

**PARALLEL ARITHMETIC AND LOGIC UNIT**

Patent Number: JP7056735  
Publication date: 1995-03-03  
Inventor(s): KATO TAKAAKI; others: 01  
Applicant(s):: NKK CORP  
Requested Patent: ☐ JP7056735  
Application Number: JP19930202207 19930816  
Priority Number(s):  
IPC Classification: G06F9/38  
EC Classification:  
Equivalents:

**Abstract**

**PURPOSE:** To provide a parallel arithmetic processor allowed to be easily loaded on normal RISC architecture and capable of executing high speed processing for a super scalar without requiring a complicated hardware circuit.  
**CONSTITUTION:** This parallel arithmetic processor is provided with decoders 11 to 14 for simultaneously decoding plural instructions, a register competition detecting circuit 20 for detecting the existence of register competition between two instructions, a PLA 31 including an instruction issue judging logic circuit, an FU use judging logic circuit, an ALU use judging logic circuit, an LD/STU use judging logic circuit, a branch instruction processing logic circuit, an execution order judging logic circuit, etc., a scheduling means 30 including a by-pass canceling circuit 32 and a memory address competition circuit 33, and an instruction issue enable/disable judging circuit 40 for determining the number of instructions to be issued based upon an instruction issue enabling/disabling information.

Data supplied from the esp@cenet database - 12

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-56735

(43) 公開日 平成7年(1995)3月3日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/38

識別記号

3 7 0 X

3 1 0 F

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数7 O L (全 30 頁)

(21) 出願番号

特願平5-202207

(22) 出願日

平成5年(1993)8月16日

(71) 出願人 000004123

日本鋼管株式会社

東京都千代田区丸の内一丁目1番2号

(72) 発明者 加藤 高明

東京都千代田区丸の内一丁目1番2号 日

本鋼管株式会社内

(72) 発明者 小野 利寿

東京都千代田区丸の内一丁目1番2号 日

本鋼管株式会社内

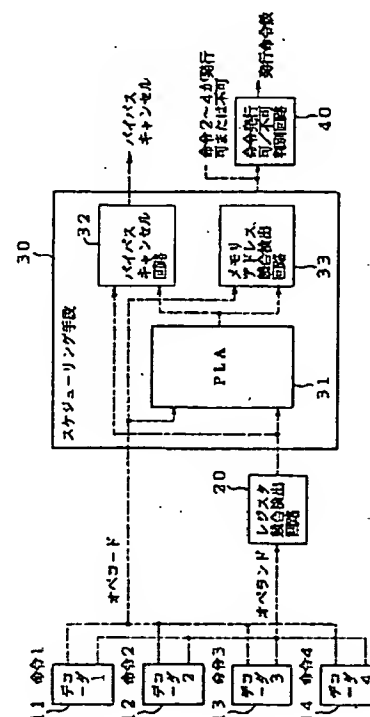
(74) 代理人 弁理士 佐々木 宗治 (外3名)

(54) 【発明の名称】 並列演算処理装置

(57) 【要約】

【目的】 スーパースカラを対象として、複雑なハードウェア回路を要さず、通常のRISCアーキテクチャ上に容易に実装可能で、しかも従来のDSよりも高速処理が可能な並列演算処理装置。

【構成】 複数の命令を同時にデコードするデコーダ11～14と、2命令間のレジスタ競合の有無を検出するレジスタ競合検出回路20と、命令発行判別論理回路、FU使用判別論理回路、ALU使用判別論理回路、LD/STU使用判別論理回路、分岐命令処理論理回路及び実行順序判別論理回路等を内蔵するPLA31、並びにバイパスキャンセル回路32及びメモリアドレス競合回路33を含むスケジューリング手段30と、命令発行の可又は不可情報により発行命令数を決める命令発行可/不可判別回路40とを備えたもの。



## 【特許請求の範囲】

【請求項1】 複数N個の命令を同時に解読し、それぞれの命令の種別情報を出力するN個のデコーダと、それぞれ演算処理命令及びロード／ストア命令を実行できる2以上前記Nまでの間の任意の数M個の命令実行ユニット(FU)と、

前記N個のデコーダで解読されたN個の命令のうちの2個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力するレジスタ競合検出回路と、

前記N個の命令の種別情報とレジスタ競合情報とを入力して、N個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及びM+1番目以降の各命令について、M+1番目までの各命令のうち命令発行可能な命令数が前記Mを越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別するFU使用判別論理回路を含むスケジューリング手段とを備えたことを特徴とする並列演算処理装置。

【請求項2】 前記M個の命令実行ユニット(FU)に代えて、同時に実行可能な、1以上前記Mまでの間の任意の数K個の演算処理ユニット(ALU)と、1以上前記Mまでの間の任意の数L個のロード／ストア処理ユニット(LD／STU)とを設け、

また前記命令実行ユニットの使用の可否を判別するFU使用判別論理回路に代えて、K+1番目以降の各命令について、K+1番目までの各命令のうち演算処理命令発行可能な命令数が前記Kを越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別するALU使用判別論理回路と、L+1番目以降の各命令について、L+1番目までの各命令のうちロード／ストア命令発行可能な命令数が前記Lを越えないか、越えるかにより前記ロード／ストア処理ユニットの使用の可否を判別するLD／STU使用判別論理回路とを設けた請求項1記載の並列演算処理装置。

【請求項3】 前記スケジューリング手段は、前記N個の命令のうち、J番目の命令について、J-1番目までの命令との間にレジスタ競合が無い場合に、J-1番目までの各命令の実行の可否にかかわらず、J番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる実行順序判別論理回路を含む請求項1または請求項2記載の並列演算処理装置。

【請求項4】 前記スケジューリング手段は、前記N個の命令のうち、J番目の命令について、該命令が分岐命令である場合に、J+1番目以降の命令を実行不可能と判別し、J+1番目以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路を含む請求項1または請求項2記載の並列演算処理装置。

【請求項5】 前記スケジューリング手段は、前記N個の命令のうち、J番目の命令について、該命令がロード／ストア命令である場合に、J-1番目までの各命令の

うちでロード／ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、J番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つJ番目の命令もロード命令であれば、J番目の命令を実行可能と判別するメモリ競合判別論理回路を含む請求項1または請求項2記載の並列演算処理装置。

10 【請求項6】 前記スケジューリング手段は、前記N個の命令のうち、J番目以降の命令について、実行可能で且つJ-1番目以前の命令より2クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを1クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する逆順序競合判別論理回路を含み、

且つ前記J-1番目までの命令のうちに前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、データ読出レジスタがJ番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、J番目以降の各命令から該命令を実行した結果情報のバイパスをキャンセルする信号を出力するバイパスキャンセル論理回路を含む請求項1または請求項2記載の並列演算処理装置。

20 【請求項7】 前記の各種論理回路は、プログラマブル・ロジック・アレイ(PLA)により実現されたものである請求項1乃至請求項6のいずれかに記載の並列演算処理装置。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明はRISC(Reduced Instruction Set Computer)プロセッサのさらなる高速化を目的としたスーパースカラのための複数命令同時発行機構を有する並列演算処理装置に関するものである。

## 【0002】

【従来の技術】図12は従来のRISCアーキテクチャのパイプライン構造を示す図であり、従来のRISCアーキテクチャは、単一のプロセッサ(CPU)を用いて、図12に示すようなパイプライン構造による時間的並列処理を導入し、CPUの命令実行に要する時間の高速化を実現したものである。図12の各命令は、次の4つのステージの処理を順次行なう。そして各ステージの処理は、それぞれ1クロックサイクルを要するので、各命令の実行開始から終了までには4クロックサイクルを要する。最初に、命令フェッチ(IF; Instruction Fetch)ステージの処理を行ない、次に、命令デコード(ID; Instruction D

## 3

ecode) ステージの処理を行ない、次に、デコード命令の実行 (EX; Execution) ステージの処理を行ない、最後に、実行結果の書き込み (または格納) (WB; Write Back) ステージの処理を行なう。

【0003】図12のパイプライン構造では、各命令は1クロックサイクルずつずらせて時間的に並列処理される。即ち命令1の実行が開始されて1クロックサイクル経過すると、命令2の実行が開始される。そして命令2の実行が開始されて1クロックサイクル経過すると、命令3の実行が開始される。このようにして、RISCでは複数の命令を並列処理できるので、理想的には図12に示すように1クロックサイクルに1命令 (CPI=1; Clock per Instruction) 実行することが可能である。しかし実際には、分岐命令やハザード (hazard、詳細は後述する) によるストール (stall; 1クロックサイクル以上のパイプラインの停止) があるため、CPIは1以上となる。

【0004】また、コンピュータの命令レベル (Instruction-level または fine-grain) の並列度を利用し、RISCの高速化を目指した並列処理アーキテクチャには、スーパーパイプライン (Superpipeline)、スーパースカラ (Superscalar)、VLIW (Very Long Instruction Word) がある。RISCの機械語命令 (オブジェクト・コード) の互換性があり、並列度の拡張性からスーパースカラが、現時点ではもっとも有望視されている。スーパースカラとは、命令の読み出しから、デコード、実行、結果の格納までの一連の処理を、複数命令に対して、同時に行なうことにより、高速処理を図るプロセッサであると定義される。そして複数命令同時発行方法の中でも、命令の Out-of-order 発行は、スーパースカラにもっとも高い性能を与えることから、現在までに多くの研究が行なわれてきた (例えば下記の参考文献のうちの [1~8] を参照)。

## 【0005】参考文献

[1] R. M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units," IBM Journal of Research and Development, Vol. 11, pp. 25-33, 1967.

[2] S. Weiss, and J. E. Smith, "Instruction Issue Logic in Pipelined Supercomputers," IEEE Trans. Comput., Vol. C-33, No. 11, pp. 1013-1022, 1984.

[3] R. D. Acosta, J. Kjelstru

## 4

p. and H. C. Torng, "An Instruction issuing approach to enhancing performance in multiple functional unit processors," IEEE Trans. Comput., Vol. C-35, pp. 815-828, 1986.

[4] W. Hwu, and Y. N. Patt, "HP Sm, A high performance restricted data flow architecture having minimal functionality," Proc. 13th Int'l Symp. Computer Architecture, pp. 297-307, 1986.

[5] G. S. Sohi, and S. Vajapeyam, "Instruction Issue Logic for High-Performance, Interruptible Pipelined Processors," Proc. 14th Int'l Symp. Computer Architecture, pp. 27-34, 1987.

[6] K. Murakami, M. Kuga, and S. Tomita, "SIMP: A Novel High-Speed Single-Processor Architecture," Proc. 16th Int'l Symp. Computer Architecture, pp. 78-85, 1989.

[7] M. D. Smith, M. S. Lam, and M. A. Horowitz, "Boosting Beyond Static Scheduling in a Superscalar Processor," Proc. 17th Int'l Symp. Computer Architecture, pp. 344-353, 1990.

[8] R. M. Keller, "Look-Ahead Processors," Computing Surveys, Vol. 7, No. 4, pp. 175-195, 1975.

[9] M. Johnson, Superscalar Microprocessor Design, Prentice Hall, 1991.

[10] J. E. Smith, and A. R. Ple szkun, "Implementation of Precise Interrupts in Pipelined Processors," Proc. 12th Int'l Symp. Computer Architecture, pp. 36-44, 1985.

[11] J. L. Hennessy, and D. A. Patterson, "Computer Architecture: A Quantitative Ap

## 5

proach, "Morgan Kaufmann Publishers, Inc., 1990.

【0006】ここで命令発行 (Instruction issue) とは、命令をパイプラインのデコードステージ (ID) から実行ステージ (EX) に動かすプロセスを言う。また Out-of-order 発行とは、命令間のデータの依存関係が無く、プログラムが正しく演算されるなら、命令の実行順序を入れ換えて実行する方式である。これに対し、プログラムの命令実行順序で命令を発行する方式を in-order 発行という。Out-of-order 発行の実現方法に関する研究の多くは、前記参考文献の [1, 2, 4~7] に記載の Tomasulo アルゴリズム及びその拡張法である。しかし、これらの方法では、レジスタ名前替えアルゴリズム (register renaming)、予約ステーション (reservation station)、タグなどの方法を組み合わせたもので、相対的に複雑なハードウェア回路を必要とし、通常の RISC アーキテクチャ上に実装する上で大きな改造を必要とするものである。

【0007】Out-of-order 発行を実現する手段には、Tomasulo アルゴリズムの他には中央命令ウィンドウ方式がある。ここで、命令ウィンドウは、フェッチ (fetch; 命令の命令キャッシュからの取り込み) された命令を一時的に格納するバッファである。この中央命令ウィンドウ方式の実現手段に下記の 3 方法がある (前記参考文献の [3, 7, 9] を参照)。

(1) Keller の原理 (参考文献の [8]) に基づく dispatch stack 法 (以下 DS と略す、参考文献 [3] を参照)

(2) register update unit 法 (以下 RUU と略す、参考文献の [7] を参照)

(3) リオーダーバッファ (reorder buffer、参考文献の [10]) を使用した方法 (参考文献の [9] を参照)

ここで、リオーダーバッファとは、分岐予測ミス時のプロセッサの状態回復のためのハードウェアであり、完了した命令による実際のレジスタの値と、未完了命令の保留代入により構成される。上記方法 (2) の RUU の性能は方法 (1) の DS の性能を下回り、同様に方法 (3) のリオーダーバッファを使用した方法も方法 (1) の DS の性能を上回ることはない (参考文献の [9] を参照)。

【0008】

【発明が解決しようとする課題】しかしながら従来の参考文献に示されたようなスーパースカラでは、複雑なハードウェア回路を必要とし、通常の RISC アーキテクチャ上に実装するには装置に大きな改造を要するという問題点があった。また Out-of-order 発行を

## 6

実現する DS (dispatch stack 法) では、レジスタに関する依存関係と、メモリに関するデータ依存関係を別機構で直列に処理するため、クロックサイクルが長くなり、結果的に処理装置の性能が低下するという問題点があった。

【0009】本発明はかかる問題点を解決するためになされたもので、スーパースカラを対象として、複雑なハードウェア回路を要さず、通常の RISC アーキテクチャ上に容易に実装可能で、しかも従来の DS よりも高速処理が可能な並列演算処理装置を得ることを目的とする。

【0010】

【課題を解決するための手段】本発明の請求項 1 に係る並列演算処理装置は、複数 N 個の命令を同時に解釈し、それぞれの命令の種別情報を出力する N 個のデコーダと、それぞれ演算処理命令及びロード/ストア命令を実行できる 2 以上前記 N までの間の任意の数 M 個の命令実行ユニット (FU) と、前記 N 個のデコーダで解釈された N 個の命令のうちの 2 個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力するレジスタ競合検出回路と、前記 N 個の命令の種別情報とレジスタ競合情報とを入力して、N 個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及び M+1 番目以降の各命令について、M+1 番目までの各命令のうち命令発行可能な命令数が前記 M を越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別する FU 使用判別論理回路を含むスケジューリング手段とを備えたものである。

【0011】本発明の請求項 2 に係る並列演算処理装置は、前記請求項 1 に係る並列演算処理装置において、前記 M 個の命令実行ユニット (FU) に代えて、同時に実行可能な、1 以上前記 M までの間の任意の数 K 個の演算処理ユニット (ALU) と、1 以上前記 M までの間の任意の数 L 個のロード/ストア処理ユニット (LD/STU) とを設け、また前記命令実行ユニットの使用の可否を判別する FU 使用判別論理回路に代えて、K+1 番目以降の各命令について、K+1 番目までの各命令のうち演算処理命令発行可能な命令数が前記 K を越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別する ALU 使用判別論理回路と、L+1 番目以降の各命令について、L+1 番目までの各命令のうちロード/ストア命令発行可能な命令数が前記 L を越えないか、越えるかにより前記ロード/ストア処理ユニットの使用の可否を判別する LD/STU 使用判別論理回路とを設けたものである。

【0012】本発明の請求項 3 に係る並列演算処理装置は、前記請求項 1 または請求項 2 に係る並列演算処理装置において、前記スケジューリング手段は、前記 N 個の命令のうち、J 番目の命令について、J-1 番目までの

命令との間にレジスタ競合が無い場合に、 $J-1$ 番目までの各命令の実行の可否にかかわらず、 $J$ 番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる実行順序判別論理回路を含むものである。

【0013】本発明の請求項4に係る並列演算処理装置は、前記請求項1または請求項2に係る並列演算処理装置において、前記スケジューリング手段は、前記 $N$ 個の命令のうち、 $J$ 番目の命令について、該命令が分岐命令である場合に、 $J+1$ 番目以降の命令を実行不可能と判別し、 $J+1$ 番目以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路を含むものである。

【0014】本発明の請求項5に係る並列演算処理装置は、前記請求項1または請求項2に係る並列演算処理装置において、前記スケジューリング手段は、前記 $N$ 個の命令のうち、 $J$ 番目の命令について、該命令がロード／ストア命令である場合に、 $J-1$ 番目までの各命令のうちでロード／ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、 $J$ 番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つ $J$ 番目の命令をロード命令であれば、 $J$ 番目の命令を実行可能と判別するメモリ競合判別論理回路を含むものである。

【0015】本発明の請求項6に係る並列演算処理装置は、前記請求項1または請求項2に係る並列演算処理装置において、前記スケジューリング手段は、前記 $N$ 個の命令のうち、 $J$ 番目以降の命令について、実行可能で且つ $J-1$ 番目以前の命令より2クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを1クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する逆順序競合判別論理回路を含み、且つ前記 $J-1$ 番目までの命令のうちで前記レジスタ競合検出回路により競合が検出されて実行できないとされた命令のうち、データ読出レジスタが $J$ 番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、 $J$ 番目以降の各命令から該当命令を実行した結果情報のバイパスをキャンセルする信号を出力するバイパスキャンセル論理回路を含むものである。

【0016】本発明の請求項7に係る並列演算処理装置は、前記請求項1乃至請求項6のいずれかに係る並列演算処理装置において、前記の各種論理回路の論理演算を実現したプログラマブル・ロジック・アレイ(PLA)を有するものである。

【0017】

【作用】本請求項1に係る発明においては、 $N$ 個のデコ

ーダは、複数 $N$ 個の命令を同時に解読し、それぞれの命令の種別情報を出力する。2以上前記 $N$ までの間の任意の数 $M$ 個の命令実行ユニット(FU)は、それぞれ演算処理命令及びロード／ストア命令を実行することができる。レジスタ競合検出回路は、前記 $N$ 個のデコーダで解読された $N$ 個の命令のうちの2個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力する。スケジューリング手段内の命令発行判別論理回路は、前記 $N$ 個の命令の種別情報とレジスタ競合情報とを入力して、 $N$ 個の命令について同時に実行でき発行可能な命令を判別し、FU使用判別論理回路は、 $M+1$ 番目以降の各命令について、 $M+1$ 番目までの各命令のうち命令発行可能な命令数が前記 $M$ を越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別する。

【0018】本請求項2に係る発明においては、前記請求項1に係る発明における前記 $M$ 個の命令実行ユニット(FU)に代えて、同時に実行可能な、1以上前記 $M$ までの間の任意の数 $K$ 個の演算処理ユニット(ALU)

と、1以上前記 $M$ までの間の任意の数 $L$ 個のロード／ストア処理ユニット(LD/STU)とが設けられ、また前記命令実行ユニットの使用の可否を判別するFU使用判別論理回路に代えて、ALU使用判別論理回路とLD/STU使用判別論理回路とが設けられる。そしてALU使用判別論理回路は、 $K+1$ 番目以降の各命令について、 $K+1$ 番目までの各命令のうち演算処理命令発行可能な命令数が前記 $K$ を越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別する。またLD/STU使用判別論理回路は、 $L+1$ 番目以降の各命令について、 $L+1$ 番目までの各命令のうちロード／ストア命令発行可能な命令数が前記 $L$ を越えないか、越えるかにより前記ロード／ストア処理ユニットの使用の可否を判別する。

【0019】本請求項3に係る発明においては、前記請求項1または請求項2に係る発明におけるスケジューリング手段に実行順序判別論理回路が含まれ、該実行順序判別論理回路は、前記 $N$ 個の命令のうち、 $J$ 番目の命令について、 $J-1$ 番目までの命令との間にレジスタ競合が無い場合に、 $J-1$ 番目までの各命令の実行の可否にかかわらず、 $J$ 番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる。

【0020】本請求項4に係る発明においては、前記請求項1または請求項2に係る発明におけるスケジューリング手段に分岐命令処理論理回路が含まれ、該分岐命令処理論理回路は、前記 $N$ 個の命令のうち、 $J$ 番目の命令について、該命令が分岐命令である場合に、 $J+1$ 番目以降の命令を実行不可能と判別し、 $J+1$ 番目以降の命令をスケジューリングの対象からはずす。

【0021】本請求項5に係る発明においては、前記請求項1または請求項2に係る発明におけるスケジューリ

ング手段にメモリ競合判別論理回路が含まれ、該メモリ競合判別論理回路は、前記N個の命令のうち、J番目の命令について、該命令がロード/ストア命令である場合に、J-1番目までの各命令のうちでロード/ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、J番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つJ番目の命令もロード命令であれば、J番目の命令を実行可能と判別する。

【0022】本請求項6に係る発明においては、前記請求項1または請求項2に係る発明におけるスケジューリング手段に逆順序競合判別論理回路及びバイパスキャンセル論理回路が含まれ、逆順序競合判別論理回路は、前記N個の命令のうち、J番目以降の命令について、実行可能で且つJ-1番目以前の命令より2クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを1クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する。またバイパスキャンセル論理回路は、前記J-1番目までの命令のうちで前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、データ読出レジスタがJ番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、J番目以降の各命令から該当命令を実行した結果情報のバイパスをキャンセルする信号を出力する。

【0023】本請求項7に係る発明においては、前記請求項1乃至請求項6に係る発明における前記の各種論理回路が、プログラマブル・ロジック・アレイ(PLA)により実現されたものである。

【0024】

【実施例】本発明はスーパースカラを対象としたものであり、同時実行可能な命令の検出と並列実行のためのハードウェア機構が必要になるが、従来の逐次型プロセッサ(スカルプロセッサ)の命令セットを変更せずに、命令レベルの並列処理を達成できるものである。本発明のアルゴリズム、ハードウェア及びPLAの実施例等を説明する前に、本発明の技術概要を従来のDSと比較して説明する。本発明は従来のDSと比較して次の4点に関して優位であるといえる。(なお詳細は後述する)。

(a) DSは、レジスタに関するデータハザードについてのみ解決を図った機構であり、FU(Functional Units; 機能ユニットもしくは命令実行ユニット、または1つのパイプライン演算機のことを指す)競合などを回避する場合には新たな機構が必要となる。このためクロック・サイクル・タイムを延ばし、性能低下を引き起こす可能性がある。一般に命令のパイプ

ライン処理を乱す原因は、ハザード(hazard)あるいは競合(dependencyまたはconflict)と呼ばれる。データハザードは命令依存関係のうち、レジスタまたはメモリに格納されたデータ相互間の依存関係のあることを指す。データハザードにはRAW(Read After Write, フロー依存; flow dependence)、WAW(Write After Write, 出力依存; output dependence)、WAR(Write After Read, 逆依存; antidependence)の各ハザードがある。FU競合(構造ハザード)回避とは、複数の命令が同時に同じFUを使用しないようにすることである。

【0025】(b) DSでは、データハザードのうち、レジスタに関する依存関係は命令発行機構(Issue Unit)によって、メモリに関するデータ依存関係は別機構(the Address Stack and the Data Unit)によって、別個に解決される。そのため、クロックサイクルが長くなってしまい、結果的に性能が低くなる。これに対し、本発明による機構は命令発行機構でレジスタ及びメモリの両方に関する依存関係を解決できるものである。DSではメモリに関する依存関係については全く考慮されていないことが性能低下の原因となっている。

(c) 本発明はWARハザード回避方法の相違によりDSの性能を上回ることができる。

(d) 本発明にはスケーラビリティ(scalability)がある。ここでスケーラビリティとは、同一アーキテクチャのもとで要素プロセッサ数をどこまで増加させるかの可能性を意味し、FU数を増減しても対応できる性質と、FUの均質性に依存しないアルゴリズムであるという特性を有する。前記FUの均質性又は不均質性とは、各FUで実行できる命令が同じであるか否かであり、FUが不均質の場合には、実行可能な命令の組み合わせによっては命令クラス・リソース競合(構造ハザード)が生じる。この構造ハザードやメモリに関するデータハザード等を並列的に解決して命令発行を行なう命令発行アルゴリズムを、PLA(Programmable Logic Array)により実現した実施例及びその効果については後述する。

【0026】最初に本発明のアルゴリズムを説明し、その後ハードウェアについての説明を行なう。アルゴリズムについては、以下の説明上の仮定を行なう。

(a) 説明上の仮定

(1) 説明を簡略化するため、同時に実行可能かどうかを扱う命令数は、4命令を基本とする。4命令はプログラム順にそれぞれ命令1(またはInst. 1, Inst. はInstructionの略とする)、命令2、命令3、命令4と呼ぶ。(なお4命令以上に拡張することは容易である。)

(2) 全ての命令は1クロックで実行される。そのため、浮動小数点演算やアドレス計算を行なうload/store命令などは考慮しない。ただし、デコード・

OP Rs1, Rs2, Rd:Rd←Rs1 OP Rs2 ..... (1)

ここで、OPは命令のオペコード(操作コード、operation code)、Rs1, Rs2はそれぞれソース(source、出所)レジスタ、Rdはディスティネーション(destination、行先)レジスタ、オペランドはソースレジスタとディスティネーションレジスタからなる。

【0027】(3) パイプライン構成は以下の(イ)～(ニ)の4ステージ構成を想定している。(イ) 命令フェッチ(IF; Instruction Fetch)ステージ、(ロ) 命令デコード(ID; Instruction Decode)ステージ、(ハ) デコード命令の実行(EX; Execution)ステージ、

(ニ) 実行結果の書き込み(または格納)(WB; Write Back)ステージである。最初に(イ)のIFステージでは、4命令を同時に命令キャッシュ(または主記憶装置)から取り込む。4命令の先頭の命令はプログラム・カウンタ(PC)から与えられる。次の

(ロ)のIDステージの前半で、4命令のオペコード(加算、減算のような操作を定義するビット列)とオペランド(演算の対象を示す。本例では、具体的にはレジスタの番地を指定する。)がそれぞれデコード(解読)される。IDステージの後半で、オペランドの値がレジ

性能向上=比較対象の実行時間/被比較対象の実行時間 ..... (3)

例えば、スーパースカラの対RISC比での性能向上を

RISCの実行時間/スーパースカラの実行時間 ..... (3')

となる。

【0029】図3は本発明とDSのID(命令デコード)ステージにおける処理方式の相違を説明する図であり、DS方式では、同時にフェッチした複数命令の並列デコード処理の次に、レジスタに関するデータハザードの回避処理、FU競合の処理、及びメモリに関するデータハザードの回避処理をそれぞれ順番に(時間的には直列処理で)解決せざるを得ない構造となっている。しかし本発明の方式では、上記の3つの処理を同時に(時間的には並列処理で)解決できる構造となっており、レジスタの読み込み処理と後述するスケジューリング処理を並列処理する。このことは、前記本発明と従来のDSとの比較した4点において、(a)、(b)項については、クロックサイクル時間tを短くすることに相当し、(c)項については、1命令の実行に要するクロックサイクル数CPIを小さくすることに相当する。

【0030】(b)本発明の命令発行は以下に示す原則に基づいてスケジュールされる。

(1) 命令1は常に発行される。

(2) 命令n(2から4)が発行される条件は、次のA～Cの3条件を満たさなければならない。すなわち、3

ロジックが少し複雑になるが、考慮することは容易である。命令形式は次の(1)式の通りである。

スタ・ファイルから読み込まれる。この読み込みと並行して命令発行のためのスケジューリングがなされる。次の(ハ)のEXステージではALU(Arithmetic unit、演算処理ユニット)命令は演算を行ない、load/store命令はデータ・キャッシュ・メモリ(Data Cache Memory)にアクセスする。最後の(ニ)のWBステージでは、EXステージの実行結果をレジスタ・ファイルに書き込む。本発明に係る上記4ステージ構成は、5段を有するパイプライン(例えば前記参考文献の[11]等)のほとんどのRISCプロセッサに適用が可能である。

【0028】(4)CPUの性能(プログラムの実行に要するCPU時間)及び性能向上については以下の

(2)式のように表せられる。

プログラムの実行に要するCPU時間

=プログラムの終了に要するクロック数×t

=IC×CPI×t

..... (2)

ここで、ICはプログラム内の命令数、CPIは1命令の実行に要するクロックサイクル数、tはクロックサイクル時間である。性能向上(speedup)はアムダール(Amdahl)の法則(前記参考文献の[11]参照)により、次の(3)式のように示される。

見る場合は、

30 条件のOR{RAW+WAW+WAR+分岐+FU競合; +は論理和(OR)を示すものとする。これらの用語の説明は後述}が成立するとき発行される。

A. (RAW, WAW, WAR条件)

命令(n-1)以前の命令と命令nとの間でデータ競合(RAW, WAW、及びWARハザード)が無いこと。

B. (分岐条件)

命令(n-1)以前の命令が分岐命令で無いこと。

C. (FU競合条件)

40 命令nが発行されると仮定して、命令nまでの命令のうち、発行される命令数が機能ユニットの数を越えないこと。

【0031】D. 本機構では、上記の競合条件を、各命令間の実行時間の差を記述したスケジューリング・パターンを用いて、あらかじめ解決しておくものである。このスケジューリング・パターンは、ウィンドウ内の命令が発行可能であるかを記述したもので、想定され得る入力パターン(レジスタ競合及び4命令の命令種別)に対し、論理演算を並列化及び圧縮化して、PLAから出力されるべきスケジューリング・パターンが実装される。

50 これによってハードウェア回路を簡易化し、かつ実行ク



ロックサイクルの短縮化を図ったものである。命令1は必ず発行されるので、命令2~4がそれぞれ発行可能であるか否かを示す3bitデータがPLAの出力となる。この最終パターンが決定するまでのスケジューリング・パターンは、各命令がそれぞれのハザードを考慮すると何クロックサイクル目で実行可能となるかを絶えず考慮している。命令1の場合を実行クロック・サイクル“1”と呼ぶ。実行クロック・サイクル“1”は言い換えると、スケジューリングの結果命令発行可能ということである。1クロックサイクル分ストールすると、実行クロック・サイクルは“2”となる。実行時にはデコード情報（レジスタ競合情報である13bit及び4命令の命令種別情報である8bitのデータ）がPLAに入力され、パターン・マッチングによってスケジューリング・パターンが出力される。

【0032】(3) 本発明のパイプライン構造におけるレジスタに関するデータハザードの回避方法について述べる。

A. RAW (Read After Write) ハザード

命令nが命令(n-1)以前の命令の結果を使って演算する場合、命令nは命令(n-1)以前の命令の実行後に実行されなければならない。そのため、命令nは命令(n-1)以前の命令とは同時に実行することはできない。これをRAWハザードと呼ぶ。図4は本発明に係るバイパス回路によるRAWハザード回避方法の説明図である。本アルゴリズムではRAWハザードが検出されると、命令nは発行されない。従って、図4のように1クロック分だけ命令の発行が待たされる。図4において、Inst. 2の発行が1クロック分待たされても、Inst. 1の結果がクロック4の前半でレジスタ・ファイルに書き込まれるのに対し、Inst. 2はクロック3の後半でレジスタからデータをすでに読み込んでしまっており、依然としてRAWハザードが生じている。この問題は、FUにバイパス(bypass; 短絡路)をつけて、出力を入力へフィードバックするフォワーディング(forwarding)技術によって解決される(前記参考文献の[11]を参照)。

【0033】B. WAW (Write After Write) ハザード

命令nの実行結果の格納場所(レジスタまたはメモリ)が、命令(n-1)以前の命令の実行結果の格納場所と同じ場合、命令nの実行結果の書き込みが命令(n-1)以前の命令の結果の書き込みより早くなされると、命令(n+1)以降の命令は間違った値を得てしまう。そこで命令nは命令(n-1)以前の命令とは同時に実行することはできない。これをWAWハザードと呼ぶ。図5は本発明に係るWAWハザード回避方法の説明図である。本アルゴリズムではWAWハザードが検出されると、命令nは発行せず、RAWハザード回避の方法と同

様にして1クロック分発行が待たされる。図5の例では、命令1~3はすべて加算命令であり、命令3は命令1及び2とRAWハザードは無いが、命令3と命令1のRdが共通のR3であり、命令1とWAWハザードがあるため、命令3の実行は1クロック分遅らされる。このような例が生じるのは、命令1のRdと命令2のRs1が共通のR3であって、命令2が命令1の結果を使用するためである。

【0034】C. WAR (Write After Read) ハザード

命令(n-1)以前の命令が、レジスタまたはメモリの値を読み込む前に、命令nがレジスタまたはメモリに実行結果を書き込んでしまうと、命令(n+1)以降の命令は間違った値を得てしまう。これをWARハザードと呼ぶ。図6は本発明に係るWARハザード現象の説明図である。図6の例では、命令2は命令1と、命令3は命令2とそれぞれRAWハザードを持つが、命令4は命令1, 2, 3のいずれともRAW及びWAWハザードを持たないため、命令4はOut-of-order効果によって命令3より実行が2クロック早まる。ところが、命令4のRdと命令3のRs2が一致するため、WARハザードとなる。しかし、1クロックのみ早まるだけでは、WARハザードが生じないのが、図6からわかる。

【0035】本発明による複数命令同時発行方式は、スケジューリング・パターン方式と称して命令ウインドウ内の4命令の実行完了がそれぞれ何クロック目でなされるかを考慮して命令発行を決定する方式である。2クロック早まるのは、図6の例すなわち、命令3・4間でしか起こり得ない。図6において命令1~4までの実行クロックサイクルは、それぞれ3, 4, 5, 3である。命令1の場合を実行クロック・サイクル“1”と呼ぶと、命令2~4は“2, 3, 1”となり、WARハザードが生じるのは、実行クロック・サイクルが(1231)というスケジュール・パターンに限定されることがわかる。本アルゴリズムでは、スケジュール・パターンが(1231)で、かつ命令4のRdと命令3のRs間のレジスタ競合があるときにWARハザードとみなす。そして、これが検出されるとスケジューリング・パターン上で命令4の発行が1クロック遅らされる。

【0036】スケジューリング・パターンにおいて、命令1は常に発行されるから、実行クロック・サイクルは“1”である。命令2は、命令1とRAW及びWAWハザードを持たなければ、命令1と同時に実行可能であり、実行クロック・サイクルは“1”だが、いずれかのハザードが存在すれば、実行クロック・サイクルは“2”となる。命令3は、命令1・2のスケジューリング・パターンが(1, 1)の時は“1”または“2”で、命令1・2のスケジューリング・パターンが(1, 2)の時は“1”、“2”または“3”ということになる。命令nの実行クロック・サイクルは“1~n”であ

り、スケジューリング・パターンが(1113)のようにクロックサイクル“2”である命令が無いというよう  
な不連続なパターンは無い。但し、Out-of-orderの効果により(1231)というパターンは存在  
する。スケジューリングの対象になる命令(命令ウィンドウ

#### 4 命令のスケジューリング・パターン

内の命令数)が4命令の場合のスケジューリング・パターンは15通りであり、これを次の表1に示す。

【0037】

【表1】

	SP	N1_4		SP	N1_4
1)	1111	4	9)	1221	2
2)	1112	3	10)	1222	1
3)	1121	3	11)	1223	1
4)	1122	2	12)	1231	2
5)	1123	2	13)	1232	1
6)	1211	3	14)	1233	1
7)	1212	2	15)	1234	1
8)	1213	2			

SP: スケジューリング・パターン

N1\_4: 発行可能な命令数

【0038】(4) 分岐命令の扱いについて

本方式では、命令(n-1)が分岐命令であったら、命令n以降の命令は取り除いてスケジューリングをする

(即ち命令n以降の発行及び実行をしない)。これは、本方式がOut-of-order命令発行方式なので、分岐命令以降の命令が分岐命令よりも先に実行されてしまうと、間違った答が得られてしまうことが頻繁に起こるからである。これについて図7を用いて説明する。図7は本発明に係る分岐命令を飛び越えてOut-of-orderが実行される場合の説明図である。図7の例では、命令2は命令1とRAWハザードを持ち、命令3は命令2の結果を使用している(RAWハザード)。命令3は分岐命令(BRT)で、命令2の実行結果をクロックサイクル4において分岐先命令のアドレスが確定し、クロックサイクル5で始めて命令フェッチが可能である。命令3は命令2の結果が格納されるレジスタR5の値で真偽が判断され、真であれば、次の命令である命令4からの4つの命令が、偽であれば、命令Xからの4つの命令がフェッチされるという命令である。命令4は分岐命令の後続命令であり、命令1~3と同時にスケジューリングされる。このとき、命令4は命令1~3とRAW, WAW, WARハザードを持たないために、Out-of-order効果によって命令3の分岐命令より前に実行される。命令3の実行結果から分岐先命令として命令Xが選択された場合、すでに命令4は

実行されており、命令4の結果がレジスタR10に書き込まれていて、レジスタR10を参照する命令Xは間違った答を得てしまう。

【0039】しかし、本方式では、スケジューリング時に分岐命令の後続の命令は無視するため、スケジューリングの対象にならず(命令ウィンドウ内には残らず)、分岐命令の結果が決定するまで命令発行はされない。従って、スケジューリング・パターン上では分岐命令の後続の命令は無意味なものとして処理され、次項で述べるFU競合やメモリに関するデータ競合(メモリアドレス競合)については対象外となる。なお、多分岐を実現するには複雑なハードウェアを必要とすること、及び多分岐を扱うプロセッサは稀であるので、本方式では対象としなかった。また、スケジューリングの対象となる命令の中で、分岐命令が検出されると、後続の命令はスケジューリングから外されるため、スケジューリング対象命令の中に分岐命令が複数あると、自動的に最初の分岐命令だけが有効となる。これに対して、分岐予測という方法がある。この方法は、上記の例において分岐予測を行ない、命令4を選択した場合に、予測した結果が正しい場合には、実行時間の大幅短縮になる。しかし予測がミスした場合には、予測した命令の実行を無効化し、パイプラインが予測前の正しいプロセッサの状態を回復する処置を要する方式である。しかし、この復元に要するハードウェア機構はかなり複雑であるため、本方式では採

用しないこととした。

【0040】(5) FU競合の回避方法について

FU競合(構造ハザード)とは、命令間にデータハザードが無く発行可能であったとしても、それを実行する適切なFUがなければ実行できないということで、これを回避しなければならない。このFU競合は、RISC、スーパーパイプライン、VLIWでは起こり得ず、スーパースカラ特有のことである。スーパースカラでもスケジュールの対象命令数とFUの数が等しければFU競合は発生しない。命令実行ユニットが $(n-1)$ 個なら、スケジューリングの対象になる命令(命令ウィンドウ内の命令数)も $(n-1)$ 個で良いように思えるが、 $(n-1)$ 個の命令が必ずしもすべて実行可能ではない。そのためいくつかの命令実行ユニットは未使用状態で、効率が良くない。そこで、スケジューリング対象命令数は、命令実行ユニット数より多いのが一般的である。この場合、 $n$ 個以上の命令が同時に実行できることがあ

り、命令を実行できるユニットが無いため、命令 $n$ は実行することができないことになる。

【0041】本発明においては、均質(Uniform)ユニット構成はALU命令とLoad/Store命令を実行でき、不均質(non-uniform)ユニット構成ではALU命令もしくはLoad/Store命令を実行できるものと想定した。いずれの場合も、その他に分岐命令の命令実行ユニットを1個有する。通常のRISCあるいはスカラプロセッサはALU、Load/Store、分岐の全ての命令実行ユニットを持っているが、常時このうち1ユニットのみが実行されると定義できる。命令ウィンドウ内の命令数が4であれば、FUユニット構成のタイプは、均質タイプがA~Cの3種、不均質タイプがD~Sの16種で合計19種となり、これを次の表2に示す。

【0042】

【表2】

## 命令数が4の時の機能ユニット構成

タイプ (均質)	Nn		Nb	FU競合 の有無	Ss
A	4		1	X	4
B	3		1	O	3
C	2		1	O	2
不均質	Nalu	Nl/s	Nb	FU競合 の有無	Ss
D	1	1	1	O	3
E	2	1	1	O	4
F	3	1	1	O	4
G	4	1	1	O	4
H	1	2	1	O	4
I	1	3	1	O	4
J	1	4	1	O	4
K	2	2	1	O	4
L	2	3	1	O	4
M	2	4	1	O	4
N	3	2	1	O	4
O	4	2	1	O	4
P	3	3	1	O	4
Q	3	4	1	O	4
R	4	3	1	O	4
S	4	4	1	X	4

Ss: 並列発行度またはスーパースカラ度

O: 考慮の必要あり、x: 考慮の必要無し

Nn: 均質ユニット構成におけるFU数

Nalu: 不均質ユニット構成におけるALU数

Nl/s: 不均質ユニット構成における  
Load/Storeユニット数

Nb: 両ユニット構成における分岐ユニット数

【0043】ユニット構成が均質の場合及び不均質の場合におけるFU競合が存在する条件は、それぞれ次の

if ( $Nn < N_{1-4}$ ),  $\rightarrow$  then FU競合 ..... (4)

if { ( $Nalu < N_{1-4}$ ) \* ( $Nl/s < N_{1-4}$ ) },

$\rightarrow$  then FU競合 ..... (5)

即ちユニット構成が均質の場合には(4)式の左側の条件が満足された場合に、またユニット構成が不均質の場合には(5)式の左側が条件が満足された場合に、それぞれFU競合が存在する。ここで、Nnは均質ユニット構成の時のFU数、Naluは不均質ユニット構成の時のALUの数、Nl/sは不均質ユニット構成の時のL

oad/Storeユニットの数、\*は論理積(AND)を示す。

【0044】表2に示されるタイプが19種あるユニット構成のうち、FU競合の有無の欄が“X”で示されるタイプAとSはFU競合が起きない。これは命令数に対してFUの数が十分にあるからである。しかし、この欄

が“○”で示される他のユニット構成については、式(4)及び(5)を基に、FU競合を考慮しなければならない。各スケジューリング・パターンと $N_{1-4}$ との関係は表1に示される。

【0045】均質ユニット構成の場合、 $N_n$ と $N_{1-4}$ を比較することによって、スケジューリング・パターンは式(4)の該当部において変えられなければならない。

一方、不均質ユニット構成の場合、 $N_{alu}$ または $N_{1/s}$ の数と $N_{1-4}$ を比較することになるが、この場合は単純な比較ではなく、 $N_{1-4}$ のうち何命令がALU命令であるか、何命令がLoad/Store命令であるかが問われる。すなわち、データハザード及び分岐命令の有無の結果、発行可能な命令のうちALU命令（もしくは

はLoad/Store命令)の数が、 $N_{alu}$ （もしくは $N_{1/s}$ ）より小さい場合は命令発行をしないようにしなければならない。これは、命令の種別に依存し、かつ実行時でないといけない。具体的には、命令 $n$ のFU競合回避は、命令 $(n-1)$ までの命令が発行可能かどうかを積算しなければならない。これをそのままハードウェア回路で実現すると、直列的な処理となるため演算に時間がかかり実行クロックサイクルが長くなるので、本発明ではこれをあらかじめ論理展開し、並列演算可能になるようにしている。FU競合を考慮した結果のスケジューリング・パターンを次の表3に示す。

【0046】

【表3】

FU競合を考慮したスケジューリング・パターン

	SP	SP1					SP	SP1	
1)	1111	1111	1211	1121	1112	8)	1213	1213	1223
		1221	1212	1122	1222	9)	1221	1221	1222
2)	1112	1112	1212	1121	1222	10)	1222	1222	
3)	1121	1121	1221	1122	1222	11)	1223	1223	
4)	1122	1122	1222			12)	1231	1231	1232
5)	1123	1123	1223			13)	1232	1232	
6)	1211	1211	1212	1221	1222	14)	1233	1233	
7)	1212	1212	1222			15)	1234	1234	

SP1: FU競合を考慮したスケジューリング・パターン

○: FU競合回避の結果、WARハザードを引き起こす可能性のあるSP(1231)に変わり得るSP1

【0047】コンパイル時にFUの競合が存在するかどうかを検出しようとすると、大幅に並列度が損なわれることになる。例えば、スケジューリング・パターンが

(1112)でタイプE(ALUユニットが2でLoad/Storeユニットが1)の場合を考えてみる。この場合 $N_{1-4}$ のうちALUが3命令以上もしくはLoad/Store命令が2命令以上であると命令3は発行できなくなるが、ALUが2命令で、Load/Store命令が1命令であれば、実行可能となる。これを判別するためには、実行時に4命令の命令種別を見て、かつこれらの $N_{alu}$ または $N_{1/s}$ の数と $N_{1-4}$ を比較しなければならない。

【0048】(6)FU競合回避によって生ずるWARハザードの回避

FU競合を考慮した結果、スケジューリング・パターンが(1231)となってしまう、WARハザードを引き起こす可能性が生じる。例えばスケジューリング・パ

ターンが(1121)であったとする。またタイプD(ALUユニットが1でLoad/Storeユニットが1)の場合で、実行時の命令種別が命令1と2がALU命令、命令4がLoad/Store命令であったとする。この場合、命令2はFU競合により発行できず、スケジューリング・パターンは(1231)に変えられなければならない。本方式では、FU競合の結果(1231)となったパターンについては再度WARハザードの有無を見ている。FU競合の結果(1221)となるパターンが次のスケジューリング時に再びFU競合を起こし、(1231)パターンとなる。この例を表3に図示した。表3において、SP1はFU競合を考慮したスケジューリング・パターンであり、このなかで楕円で囲まれたものは、FU競合回避の結果、WARハザードを引き起こす可能性のあるSP(1231)に変わり得るSP1である。これまでスケジューリング・パターンは実行サイクルクロック数を考慮してきたが、ハードウェア

回路として必要なのは発行できるか (1) 否か (0) である。ここで出力されるスケジューリング・パターンは、次の表 4 の SP 2 に示すようなスケジューリング・

パターンが PLA の出力となる。

【0049】

【表 4】

4 命令の出力 スケジューリング・パターン

	SP	SP2	in-order/ out-of-order		SP	SP2	in-order/ out-of-order
1)	1111	1111	in-order	9)	1221	1001	out-of-order
2)	1112	1110	in-order	10)	1222	1000	in-order
3)	1121	1101	out-of-order	11)	1223	1000	in-order
4)	1122	1100	in-order	12)	1231	1001	out-of-order
5)	1123	1100	in-order	13)	1232	1000	in-order
6)	1211	1011	out-of-order	14)	1233	1000	in-order
7)	1212	1010	out-of-order	15)	1234	1000	in-order
8)	1213	1010	out-of-order				

SP2: 出力スケジューリング・パターン (1: 命令発行する, 0: 命令発行せず)

【0050】表 4 において、SP 2 の欄は 4 命令の出力スケジューリング・パターンを示し、欄内の (1) は命令を発行できる場合を、(0) は命令を発行できない場合をそれぞれ示している。ここで、注意を要することがある。図 6 の例においてレジスタに関する WAR ハザードは命令 4 が 2 クロックではなく、1 クロックのみ早く実行されることによって回避される。しかし、1 クロック早く実行されると RAW ハザードとみなされ、バイパス回路が開かれ WAR ハザードと同じ現象となり、命令 3 は間違った答を得てしまう。このような WAR ハザード条件はスケジューリング・パターン上で回避できる

が、この場合は、バイパス回路を開かないようにして並列度を損なうことなく回避することができ、これをバイパス・キャンセル方式と呼ぶ。Out-of-order 発行された場合で、かつ Out-of-order になった命令間で次のクロックに命令の逆順による RAW ハザードを引き起こすことになるレジスタ競合がある場合に、バイパス回路はキャンセルされなければならない。この条件を次の表 5 に示す。

30 【0051】

【表 5】

## WAR ハザード回避のための バイパス回路 をキャンセルする条件

	SP2	条件	閉じられるバイパス回路
1)	1111	条件無し	命令3のRs1またはRs2
2)	1110	条件無し	
3)	1101	AND (rc5 OR rc6)	
4)	1100	条件無し	
5)	1100	条件無し	
6)	1011	AND (rc1 OR rc2)	命令2のRs1またはRs2
	1011	AND (rc3 OR rc4)	命令2のRs1またはRs2
	1011	AND (rc1 OR rc2) AND (rc3 OR rc4)	命令2のRs1またはRs2
			命令3のRs1またはRs2
7)	1010	AND (rc1 OR rc2)	命令2のRs1またはRs2
8)	1010	AND (rc1 OR rc2)	命令2のRs1またはRs2
9)	1001	AND (rc3 OR rc4)	命令2のRs1またはRs2
	1001	AND (rc5 OR rc6)	命令3のRs1またはRs2
10)	1000	条件無し	命令2のRs1またはRs2 命令3のRs1またはRs2
11)	1000	条件無し	
12)	1001	AND (rc3 OR rc4)	
	1001	AND (rc5 OR rc6)	
13)	1000	条件無し	
14)	1000	条件無し	命令2のRs1またはRs2 命令3のRs1またはRs2
15)	1000	条件無し	

rc1: 命令3のRdと命令2のRs1間のレジスタ競合

rc2: 命令3のRdと命令2のRs2間のレジスタ競合

rc3: 命令4のRdと命令2のRs1間のレジスタ競合

rc4: 命令4のRdと命令2のRs2間のレジスタ競合

rc5: 命令4のRdと命令3のRs1間のレジスタ競合

rc6: 命令4のRdと命令3のRs2間のレジスタ競合

【0052】表5のSP2の欄における数字の“1”は命令を実行する場合を、“0”は命令を実行しない場合をそれぞれ示している。これに対し、DS方式では各命令が何クロック目で実行されるかということには着目せず、RAW、WAW及びWARハザードを引き起こす可能性のあるレジスタ競合の論理和（OR）を取り、これらのレジスタ競合が一つもなければ命令が発行される。この方法の良さはレジスタに関するデータハザードだけに関して言えば、ハードウェア機構が単純化されることである。しかし、性能に関して言えば、本発明のスケジューリング・パターン方式の方がDS方式より良い。これは、上記に述べたようにWARハザードを引き起こす可能性のあるレジスタ競合が存在しても、後続の命令が1クロックのみ実行が早まるだけではWARハザードは

生じないということによる。この典型例を図8に示す。

【0053】図8は本発明に係るWARハザード回避方法及びその効果の説明図である。図8の例では、命令2が命令1とRAWハザードを持つため実行が1クロック遅らされる。命令3は命令1、2とRAW及びWAWハザードを持たないが、WARハザードを引き起こす可能性のあるレジスタ競合が存在する。この場合に本発明の方式では、命令3は命令2に比べて1クロックのみ実行が早まるので、実際はWARハザードではないが、DS方式では各命令の実行クロック数を見ていないために、この例のようにWARハザードを引き起こす可能性のあるレジスタ競合が生じる限り、WARハザードとみなさざるを得ない。本方式はDS方式に比べ図8の（a）、（b）に示したように命令の実行時間が短縮される。

【0054】(7) メモリに関するデータハザード（メモリアドレス競合）の回避方法について

マルチポート・キャッシュ・メモリを使用すれば、命令ウィンドウ内の複数のLoad/Store命令が同時実行可能である。しかし、これらの命令がOut-of-order発行されると、レジスタに関するデータハザードが無くても、メモリに関するデータハザードを引き起こすことがある。このメモリに関するデータハザードは、これらの命令が参照するレジスタは異なっても、レジスタが指し示すアドレスが一致することにより生じ、命令の実行時でないとわからない。これには次の3つのケースが存在する。

- ・ Load命令の後のStore命令がOut-of-order発行されると、WARハザードが起きる可能性がある。

- ・ Store命令の後のLoad命令がOut-of-

order発行されると、RAWハザードが起きる可能性がある。

- ・ Store命令の後のStore命令がOut-of-order発行されると、WAWハザードが起きる可能性がある。

それゆえ、2命令以上のLoad/Store命令がOut-of-order発行される場合、命令発行される命令がLoad命令だけに限られる場合を除いて、in-order発行に変えられなければならない。この場合に対象となるのは、表4の（SP2）に示す出力スケジューリング・パターンのうちOut-of-order発行となるものであり、実行時に命令2～4の命令種別に基づいてハードウェア回路によって変えられる。変えられるべき条件を次の表6の（SP3）に示す。

【0055】

【表6】



メモリに関するデータハザードを考慮した  
スケジューリング・パターン

	SP2	変更されるべき Load/Store 命令の組合せ条件	SP3
1)	1111	変更無し	1111
2)	1110	変更無し	1110
3)	1101	$(y0 \text{ AND } z1) \text{ OR } (y1 \text{ AND } z0) \text{ OR } (y1 \text{ AND } z1)$	1100
4)	1100	変更無し	1100
5)	1100	変更無し	1100
6)	1011	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1001
	1011	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1010
	1011	$(x0 \text{ AND } y1 \text{ AND } z0) \text{ OR } (x0 \text{ AND } y1 \text{ AND } z1) \text{ OR } (x1 \text{ AND } y0 \text{ AND } z1) \text{ OR } (x1 \text{ AND } y1 \text{ AND } z0) \text{ OR } (x1 \text{ AND } y1 \text{ AND } z1)$	1000
7)	1010	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1000
8)	1010	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1000
9)	1001	$(x0 \text{ AND } z1) \text{ OR } (x1 \text{ AND } z0) \text{ OR } (x1 \text{ AND } z1)$	1000
	1001	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1000
10)	1000	変更無し	1000
11)	1000	変更無し	1000
12)	1001	$(x0 \text{ AND } z1) \text{ OR } (x1 \text{ AND } z0) \text{ OR } (x1 \text{ AND } z1)$	1000
	1001	$(x0 \text{ AND } y1) \text{ OR } (x1 \text{ AND } y0) \text{ OR } (x1 \text{ AND } y1)$	1000
13)	1000	変更無し	1000
14)	1000	変更無し	1000
15)	1000	変更無し	1000

SP3: メモリに関するデータハザードを考慮した 変更後の  
スケジューリング・パターン

$x0: \text{Inst.2} = \text{Load}, \quad x1: \text{Inst.2} = \text{Store}, \quad y0: \text{Inst.3} = \text{Load}, \quad y1: \text{Inst.3} = \text{Store},$   
 $z0: \text{Inst.4} = \text{Load}, \quad z1: \text{Inst.4} = \text{Store},$

【0056】次に本発明の複数命令同時発行機構について説明する。

(a) ハードウェア

本発明のハードウェア構成を図面を用いて説明する。図2は本発明に係るスーパースカラ・アーキテクチャを示すブロック図である。同図において、1は命令デコード及びスケジューリング機構、2はプログラム・カウンタ(PC)及び分岐機構、3は命令キャッシュ及びフェッチ機構、4は機能ユニット(FU)、5はデータキャッシュ、6はレジスタ・ファイルであり、本アーキテクチャは主に上記の6つの部分から構成される。均質ユニッ

構成の場合、各機能ユニットはALU命令とLoad/Store命令を実行することができる。不均質ユニット構成の場合、機能ユニットはALU命令もしくはLoad/Store命令を実行することができる。Load/Storeユニットだけがデータキャッシュにアクセスされる。均質又は不均質ユニットのいずれの構成においても分岐命令は分岐機構で処理される。

【0057】図1は図2の命令デコード及びスケジューリング機構についての詳細な構成図である。本機構は4つのデコード11~14、レジスタ競合検出回路20、スケジューリング手段30及び命令発行可/否判別回路

40から構成される。またスケジューリング手段30には、PLA31、バイパスキャンセル回路32及びメモリアドレス競合検出回路33が含まれている。PLA31には、複数の命令のうち同時に実行でき発行可能な命令を判別する命令発行判別論理回路、複数のFUの使用の可否を判別するFU使用判別論理回路、複数のALUの使用の可否を判別するALU使用判別論理回路、複数のロード/ストア処理ユニット(LD/STU)の使用の可否を判別するLD/STU使用判別論理回路、分岐命令以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路、命令の実行順序を入れ換えて実行できる可否かを判別する実行順序判別論理回路などの各種論理回路が含まれている。

【0058】また図1においては、バイパスキャンセル回路32とメモリアドレス競合検出回路33は、比較的まとまりやすい回路なので、独立の回路構成とした例を示したが、これらの回路をPLA31に含ませた構成とすることも可能である。本発明では、このように命令発行に係る各種論理回路をPLA(Programmable Logic Array)のようなハードウェア装置を用いて実現されるため、CPUの数を増やすなどの装置の構成が変更されても、PLA内のアルゴリズムをソフトウェア的に変更するだけで、他のハードウェア要素をいっさい変えずに済むという特徴がある。バイパスキャンセル回路32は、命令の実行順序を逆にしてレジスタ競合が有る場合に、該当命令を実行した結果情報のバイパスをキャンセルする回路である。命令発行可/不可判別回路40は、スケジューリング手段30内のPLA31及びメモリアドレス競合検出回路33を介して、出力される命令2~4の発行可または発行不可の判別信号により発行命令数を決める回路である。

【0059】本発明の命令デコード及びスケジューリングの操作は、次の7段階に分かれる。

(1) ウィンドウ内の4命令は、それぞれ同時に独立にデコードされ、オペコード及びオペランドが解読される。4命令のオペランドを比較することによって命令間のレジスタ競合が13bitのデータに基づきレジスタ競合検出回路20で検出される。オペコードのデコード結果は、4種類の命令(ALU、Load、Store、そして分岐命令の4種類)に分類される。各命令は2bitで記述され、合計8bitである。

(2) デコードが完了すると、ソースレジスタの値がレジスタ・ファイルから読み込まれる。レジスタ読み込みと並行してスケジューリングが以下のようになされる(図3参照)。

(3) レジスタ競合と命令種別の情報(前記13bitと8bitの合計21bitのデータ)がPLAに入力される。分岐命令の有無やウィンドウ内での位置は、命令種別データより同定される。パターン・マッチングの結果、PLAからスケジューリング・パターンが出力さ

れる。この出力は命令2~4がそれぞれ発行可能であるか否かを示すもので、3bitのデータである。このスケジューリング・パターンと命令種別とが前記表6の条件を満足する場合、メモリアドレス競合とみなされ、該当する命令の発行は取り止められる。表6の論理を実現したものが図1のメモリアドレス競合検出回路33である。

【0060】(4) データバス選択部で、FUのディスティネーションレジスタと発行される命令のソースレジスタとの間にRAWハザードが検出されると、レジスタ・ファイルから読み出された値の代わりにバイパスされた値を選択する(【0032】の(3)A項[RAWハザード]を参照)。

(5) 発行可能な命令のオペコードとソースレジスタとディスティネーションレジスタのそれぞれの値が適切なFUに送られる。

(6) WARハザードを引き起こす可能性のあるレジスタ競合とスケジューリング・パターンとの組合せが検出されたなら、バイパス回路キャンセル回路32においてバイパス回路は不能にされる。

(7) ウィンドウから実際に発行された命令の数だけ、スケジューリングのため次の命令をフェッチしなければならない。次にフェッチされる命令のアドレスを保持する。PCは直前に発行された命令数に依存して決定されるため、スケジューリング・パターンからどの命令が発行されたかに基づいて算出される。

【0061】(b) スケジューリング・パターン及びPLAの設計

スケジューリング・パターンはPLAのハードウェア・サイズが小さくなるように設計される。PLAへの入力は前記21bitであり、その内訳は命令間のレジスタ競合に13bit、命令種別の解読に8bitである。これらの詳細を以下に示す。

(1) 命令間のレジスタ競合(13bit)

・RAW hazardに関するレジスタ競合(6bit); e, f, g, h, i, j

・WAW hazardに関するレジスタ競合(6bit); k, l, m, n, o, p

・WAR hazardに関するレジスタ競合(1bit); q

ここで、e: 命令1・2間のRd-Rsレジスタ競合(RAW)、

f: 命令1・3間のRd-Rsレジスタ競合(RAW)、

g: 命令2・3間のRd-Rsレジスタ競合(RAW)、

h: 命令1・4間のRd-Rsレジスタ競合(RAW)、

i: 命令2・4間のRd-Rsレジスタ競合(RAW)、

50 w)、

j : 命令 3・4 間の R d - R s レジスタ競合 (R A W) 、  
 k : 命令 1・2 間の R d - R d レジスタ競合 (W A W) 、  
 l : 命令 1・3 間の R d - R d レジスタ競合 (W A W) 、  
 m : 命令 2・3 間の R d - R d レジスタ競合 (W A W) 、  
 n : 命令 1・4 間の R d - R d レジスタ競合 (W A W) 、  
 o : 命令 2・4 間の R d - R d レジスタ競合 (W A W) 、  
 p : 命令 3・4 間の R d - R d レジスタ競合 (W A W) 、  
 q : 命令 4・3 間の R s - R d レジスタ競合 (W A R) である。

#### 【0062】(2) 命令解読 (8 b i t)

各命令はそれぞれ次の 2 b i t で記述され、合計 8 b i t である。

ALU 命令 …… 1 0、  
 LOAD 命令 …… 0 0、  
 STORE 命令 …… 0 1、  
 BRANCH 命令 …… 1 1

命令 1～4 までのデコード結果 (各 2 b i t) を次のように表記する。

命令 1 の上位ビット… d 1 1、命令 1 の下位ビット… d 1 0、  
 命令 2 の上位ビット… d 2 1、命令 2 の下位ビット… d 2 0、

命令 3 の上位ビット… d 3 1、命令 3 の下位ビット… d 3 0、  
 命令 4 の上位ビット… d 4 1、命令 4 の下位ビット… d 4 0、

#### 【0063】(3) レジスタに関するデータハザードの回避手順

A. 上記の 13 b i t のレジスタ競合データを基にレジスタに関するデータハザードを考慮してスケジューリング・パターンを作成する。この結果のスケジューリング・パターンが前記表 1 に集約される。

B. 分岐命令を考慮してスケジューリング・パターンが変更される。具体的には命令 1 が分岐命令であれば、命令 2 以降の命令が、命令 2 が分岐命令であれば、命令 3 以降の命令が、命令 3 が分岐命令であれば、命令 4 がそれぞれ自動的に発行不可とされる。

C. F U 競合を考慮することによって表 3 の S P 1 のようなスケジューリング・パターンに変更される。

D. F U 競合回避によって生じる W A R ハザードを表 3 の楕円印に着目して回避して、スケジューリング・パターンは P L A の出力として表 4 の S P 2 のような形でまとめられる。

E. これらの論理を圧縮し、共通項を抽出することによって実際の P L A に搭載される積項 (p r o d u c t t e r m s) は次の表 7 に示すように 19～43 b i t でかなり小さくなり、妥当なハードウェア・サイズと言える。

#### 【0064】

#### 【表 7】

35

36

各ユニット構成におけるPLA上の積項数

タイプ 均質	命令 2					命令 3					命令 4					合計
	RAW MAY	分岐	FUC	MAR	小計	RAW MAY	分岐	FUC	MAR	小計	RAW MAY	分岐	FUC	MAR	小計	
A	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	X(0)	O(4)	11	19
B	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
C	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
不均質																
D	O(2)	O(1)	O(2)	X(0)	5	O(4)	O(1)	O(4)	X(0)	9	O(6)	O(1)	O(6)	O(16)	29	43
E	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(3)	X(0)	8	O(6)	O(1)	O(5)	O(9)	21	33
F	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(9)	20	31
G	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(3)	O(9)	19	30
H	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(3)	X(0)	8	O(6)	O(1)	O(5)	O(9)	21	33
I	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(9)	20	31
J	O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(3)	O(9)	19	30
K	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(4)	15	25
L	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(3)	O(4)	14	23
M	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
N	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(3)	O(4)	14	23
O	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
P	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(2)	O(4)	13	21
Q	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
R	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
S	O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	X(0)	O(4)	11	19

O: 考慮の必要あり、X: 考慮の必要無し、( ): 積項の数、FUC: FU競合

【0065】表7において、ユニット構成によって積項が異なるのはFU競合回避方法が異なるためである。また表7に示すように積項の数を減少させることによって、本スケジューリング・パターン方式が小さなハードウェア構成で実現でき、FUユニット構成が変わっても、PLA内のアルゴリズムをソフトウェア的に変更するだけで、他のハードウェアはまったく変えることなく、複数命令をOut-of-orderで同時に発行できることがわかる。

【0066】本発明のPLAの一実施例を図面を用いて説明する。図9～図11は、それぞれユニット構成がタイプA、B、H(表2参照)の場合のPLA上の命令発行論理回路図である。

#### PLAの実施例1

タイプA(均質ユニットが4つ)の場合の実施例

図9はタイプAのユニット構成の場合のPLA上の命令

発行論理回路図である。まず、命令2の発行に支障があるかどうかは図9の(a)の回路により、命令1のディスティネーションレジスタと命令2のソースレジスタが同一であるか(e)、命令1と命令2のディスティネーションレジスタが同一であるか(k)、または、命令1が分岐命令( $d11=1$ ,  $d10=1$ , 従って $d11 * d10=1$ )の場合に、図9の(a)の回路の出力が1となり発行に支障がある。それ以外では、出力は0となり支障無しとなり発行される。

【0067】次に、命令3の発行に支障があるかどうかは図9の(b)の回路により、命令1のディスティネーションレジスタと命令3のソースレジスタが同一であるか(f)、命令2のディスティネーションレジスタと命令3のソースレジスタが同一であるか(g)、命令1と命令3のディスティネーションレジスタが同一であるか(1)、命令2と命令3のディスティネーションレジ

タが同一であるか (m)、または、命令 1 か命令 2 が分岐命令 ( $d11 * d10 = 1$  または  $d21 * d20 = 1$ ) の場合に、図 9 の (b) の回路の出力が 1 となり発行に支障がある。それ以外では、出力は 0 となり支障無しとなり発行される。

【0068】最後に、命令 4 の発行に支障があるかどうかは図 9 の (c) の回路により、命令 1 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (h)、命令 2 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (i)、命令 3 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (j)、命令 1 と命令 4 のディスティネーションレジスタが同一であるか (n)、命令 2 と命令 4 のディスティネーションレジスタが同一であるか (o)、命令 3 と命令 4 のディスティネーションレジスタが同一であるか (p)、または、命令 1 か命令 2 か命令 3 が分岐命令 ( $d11 * d10 = 1$  または  $d21 * d20 = 1$  または  $d31 * d30 = 1$ ) であるか、または、命令 1 のディスティネーションレジスタと命令 2 のソースレジスタが同一であるか (e)、命令 1 と命令 2 のディスティネーションレジスタが同一であり (k)、かつ命令 2 のディスティネーションレジスタと命令 3 のソースレジスタが同一であるか (g)、命令 2 と命令 3 のディスティネーションレジスタが同一であり (m)、かつ命令 4 のディスティネーションレジスタと命令 3 のソースレジスタが同一である場合 (q) に、図 9 の

(c) の回路の出力が 1 となり発行に支障がある。それ以外では、出力は 0 となり支障無しとなり発行される。

#### 【0069】PLAの実施例 2

タイプ B (均質ユニットが 3 つ) の場合の実施例

図 10 はタイプ B のユニット構成の場合の PLA 上の命令発行論理回路図である。まず、命令 2 の発行に支障があるかどうかは図 10 の (a) の回路により、命令 1 のディスティネーションレジスタと命令 2 のソースレジスタが同一であるか (e)、命令 1 と命令 2 のディスティネーションレジスタが同一であるか (k)、または、命令 1 が分岐命令 ( $d11 = 1$ ,  $d10 = 1$ , 従って  $d11 * d10 = 1$ ) の場合に、図 10 の (a) の回路の出力が 1 となり発行に支障がある。それ以外では、出力は 0 となり支障無しとなり発行される。

【0070】次に、命令 3 の発行に支障があるかどうかは図 10 の (b) の回路により、命令 1 のディスティネーションレジスタと命令 3 のソースレジスタが同一であるか (f)、命令 2 のディスティネーションレジスタと命令 3 のソースレジスタが同一であるか (g)、命令 1 と命令 3 のディスティネーションレジスタが同一であるか (l)、命令 2 と命令 3 のディスティネーションレジスタが同一であるか (m)、または、命令 1 か命令 2 が分岐命令 ( $d11 * d10 = 1$  または  $d21 * d20 = 1$ ) の場合に、図 10 の (b) の回路の出力が 1 となり

発行に支障がある。それ以外では、出力は 0 となり支障無しとなり発行される。

【0071】最後に、命令 4 の発行に支障があるかどうかは図 10 の (c) の回路により、命令 1 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (h)、命令 2 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (i)、命令 3 のディスティネーションレジスタと命令 4 のソースレジスタが同一であるか (j)、命令 1 と命令 4 のディスティネーションレジスタが同一であるか (n)、命令 2 と命令 4 のディスティネーションレジスタが同一であるか (o)、命令 3 と命令 4 のディスティネーションレジスタが同一であるか (p)、または、命令 1 か命令 2 か命令 3 が分岐命令 ( $d11 * d10 = 1$  または  $d21 * d20 = 1$  または  $d31 * d30 = 1$ ) であるか、

【0072】または、命令 1 のディスティネーションレジスタと命令 2 のソースレジスタが同一で無く (! e、e の論理反転)、かつ命令 1 と命令 2 のディスティネーションレジスタが同一で無く (! k)、かつ命令 1 のディスティネーションレジスタと命令 3 のソースレジスタが同一で無く (! f)、かつ命令 2 のディスティネーションレジスタと命令 3 のソースレジスタが同一で無く (! g)、かつ命令 1 と命令 3 のディスティネーションレジスタが同一で無く (! l)、かつ命令 2 と命令 3 のディスティネーションレジスタが同一で無いか (!

m)、または、命令 1 のディスティネーションレジスタと命令 2 のソースレジスタが同一であるか (e)、命令 1 と命令 2 のディスティネーションレジスタが同一であり (k) かつ命令 2 のディスティネーションレジスタと命令 3 のソースレジスタが同一であるか (g)、命令 2 と命令 3 のディスティネーションレジスタが同一であり (m) かつ命令 4 のディスティネーションレジスタと命令 3 のソースレジスタが同一である場合 (q) に、図 10 の (c) の回路の出力が 1 となり発行に支障がある。それ以外では、出力は 0 となり支障無しとなり発行される。

#### 【0073】PLAの実施例 3

タイプ H (不均質ユニット構成、ALU が 1 つと Load/store ユニットが 2 つ) の場合の実施例

図 11 はタイプ H のユニット構成の場合の PLA 上の命令発行論理回路図である。まず、命令 2 の発行に支障があるかどうかは図 11 の (a) の回路により、命令 1 のディスティネーションレジスタと命令 2 のソースレジスタが同一であるか (e)、命令 1 と命令 2 のディスティネーションレジスタが同一であるか (k)、または、命令 1 が分岐命令 ( $d11 = 1$ ,  $d10 = 1$ ) であるか ( $d11 * d10 = 1$ )、または、命令 1 が ALU 命令 ( $d11 = 1$ ,  $d10 = 0$ ) でかつ命令 2 も ALU 命令 ( $d21 = 1$ ,  $d20 = 0$ ) の場合に、図 11 の (a) の回路の出力が 1 となり発行に支障がある。それ以外で

は、出力は0となり支障無しとなり発行される。

【0074】次に、命令3の発行に支障があるかどうかは図11の(b)の回路により、命令1のディスティネーションレジスタと命令3のソースレジスタが同一であるか(f)、命令2のディスティネーションレジスタと命令3のソースレジスタが同一であるか(g)、命令1と命令3のディスティネーションレジスタが同一であるか(l)、命令2と命令3のディスティネーションレジスタが同一であるか(m)、または、命令1か命令2が分岐命令( $d11 * d10 = 1$ または $d21 * d20 = 1$ )であるか、または、命令1がALU命令( $d11 = 1, d10 = 0$ )でかつ命令3もALU命令( $d31 = 1, d30 = 0$ )であるか、または、命令2がALU命令( $d21 = 1, d20 = 0$ )であってかつ命令3もALU命令( $d31 = 1, d30 = 0$ )でありかつ、命令1のディスティネーションレジスタと命令2のソースレジスタが同一で無く(!e)、かつ命令1と命令2のディスティネーションレジスタが同一で無いか(!k)、または、命令1がLoad/store命令( $d11 = 0$ )かつ、命令2がLoad/store命令で( $d21 = 0$ )かつ、命令3がLoad/store命令で( $d31 = 0$ )かつ、命令1のディスティネーションレジスタと命令2のソースレジスタが同一で無く(!e)、かつ命令1と命令2のディスティネーションレジスタが同一で無い(!k)場合に、図11の(b)の回路の出力が1となり発行に支障がある。それ以外では、出力は0となり支障無しとなり発行される。

【0075】最後に、命令4の発行に支障があるかどうかは図11の(c)の回路により、命令1のディスティネーションレジスタと命令4のソースレジスタが同一であるか(h)、命令2のディスティネーションレジスタと命令4のソースレジスタが同一であるか(i)、命令3のディスティネーションレジスタと命令4のソースレジスタが同一であるか(j)、命令1と命令4のディスティネーションレジスタが同一であるか(n)、命令2と命令4のディスティネーションレジスタが同一であるか(o)、命令3と命令4のディスティネーションレジスタが同一であるか(p)、または、命令1か命令2か命令3が分岐命令( $d11 * d10 = 1$ または $d21 * d20 = 1$ または $d31 * d30 = 1$ )であるか、または、命令1がALU命令( $d11 = 1, d10 = 0$ )でかつ命令4もALU命令( $d41 = 1, d40 = 0$ )であるか、または、命令2がALU命令( $d21 = 1, d20 = 0$ )であってかつ命令4もALU命令( $d41 = 1, d40 = 0$ )でありかつ、命令1のディスティネーションレジスタと命令2のソースレジスタが同一で無く(!e)、かつ命令1と命令2のディスティネーションレジスタが同一で無いか(!k)、

【0076】または、命令3がALU命令( $d31 = 1, d30 = 0$ )であってかつ命令4もALU命令( $d$

$d41 = 1, d40 = 0$ )でありかつ、命令1のディスティネーションレジスタと命令3のソースレジスタが同一で無く(!f)、かつ命令2のディスティネーションレジスタと命令3のソースレジスタが同一で無く(!g)、かつ命令1と命令3のディスティネーションレジスタが同一で無く(!l)、かつ命令2と命令3のディスティネーションレジスタが同一で無いか(!m)、または、命令2がLoad/store命令で( $d21 = 0$ )かつ、命令4がLoad/store命令で( $d41 = 0$ )かつ、命令1のディスティネーションレジスタと命令2のソースレジスタが同一で無く(!e)、かつ命令1と命令2のディスティネーションレジスタが同一で無いか(!k)、または、命令1がLoad/store命令で( $d11 = 0$ )かつ、命令3がLoad/store命令で( $d31 = 0$ )かつ、命令4がLoad/store命令で( $d41 = 0$ )かつ、命令1のディスティネーションレジスタと命令3のソースレジスタが同一で無く(!f)、かつ命令2のディスティネーションレジスタと命令3のソースレジスタが同一で無く(!g)、かつ命令1と命令3のディスティネーションレジスタが同一で無く(!l)、かつ命令2と命令3のディスティネーションレジスタが同一で無いか(!m)、または、下記の信号(i)、及び(ロ)が共に1で、かつ命令4のディスティネーションレジスタと命令3のソースレジスタが同一である場合(q)に、図11の(c)の回路の出力が1となり発行に支障がある。それ以外では、出力は0となり支障無しとなり発行される。

【0077】但し、信号(i)は、命令1のディスティネーションレジスタと命令2のソースレジスタが同一であるか(e)、命令1と命令2のディスティネーションレジスタが同一であるか(k)、または命令1がALU命令( $d11 = 1, d10 = 0$ )でかつ命令2もALU命令( $d21 = 1, d20 = 0$ )である場合に、1となる。また信号(ロ)は、命令2のディスティネーションレジスタと命令3のソースレジスタが同一であるか

(g)、命令2と命令3のディスティネーションレジスタが同一であるか(m)、または命令2がALU命令( $d21 = 1, d20 = 0$ )でかつ命令3もALU命令( $d31 = 1, d30 = 0$ )である場合に、1となる。

【0078】なお、本発明は、スーパースカラを対象とするものではあるが、必ずしもこれに限定されるものではなく、パイプライン化された演算機が複数あれば、スーパースカラに限らずスーパーコンピュータなどにも適用可能である。また、対象はRISCに限定されず、CISC(Complex Instruction Set Computer)プロセッサに対しても実装は容易である。

【0079】

【発明の効果】以上のように本発明によれば、複数N個の命令を同時に解読し、それぞれの命令の種別情報を出

力するN個のデコーダと、それぞれ演算処理命令及びロード／ストア命令を実行できる2以上前記Nまでの間の任意の数M個の命令実行ユニット(FU)と、前記N個のデコーダで解読されたN個の命令のうちの2個の命令間のレジスタ競合情報を出力するレジスタ競合検出回路と、N個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及びM+1番目以降の各命令について、前記命令実行ユニットの使用の可否を判別するFU使用判別論理回路を含むスケジューリング手段とを備えるようにしたので、RISCなどのスカラプロセッサのオブジェクト・コード(機械語命令)の互換性を維持しつつ、コンパイラなどのソフトウェアの変更をいっさい行わずに、通常のRISCアーキテクチャ上に実装可能となり、複数命令を同時に発行して実行することにより、RISCなどのスカラプロセッサの高速化・高性能化が可能となる。

【0080】また本発明によれば、前記M個の命令実行ユニット(FU)に代えて、同時に実行可能な、1以上前記Mまでの間の任意の数K個の演算処理ユニット(ALU)と、1以上前記Mまでの間の任意の数L個のロード／ストア処理ユニット(LD/STU)とを設け、また前記FU使用判別論理回路に代えて、K+1番目以降の各命令について、前記演算処理ユニットの使用の可否を判別するALU使用判別論理回路と、L+1番目以降の各命令について、前記ロード／ストア処理ユニットの使用の可否を判別するLD/STU使用判別論理回路とを設けるようにしたので、表2で示した不均質ユニット構成の場合にも、FU競合の有無を判別し、複数命令を同時に発行して実行することにより、同様にスカラプロセッサの高速化が可能となる。

【0081】また本発明によれば、スケジューリング手段内に、命令の実行順序を入れ換えて実行の可否を判別する実行順序判別論理回路を設けるようにしたので、命令間のデータの依存関係が無く、プログラムが正しく演算され得る場合には、命令の実行順序を入れ換えて実行するOut-of-orderを実施することが可能となり、スカラプロセッサの性能が大幅に向上した。

【0082】また本発明によれば、スケジューリング手段内に、分岐命令を識別すると、それ以降の命令をスケジューリングの対象からはずす分岐命令処理判別論理回路を設けるようにしたので、従来の分岐予測法における予測ミスの場合の復元に要する複雑なハードウェア機構は不要となり、ハードウェアが簡略化された。

【0083】また本発明によれば、スケジューリング手段内に、命令がロード／ストア命令である場合にメモリ競合を判別するメモリ競合判別論理回路を設けるようにしたので、パイプライン内の命令の流れを乱す前記レジスタに関するデータハザード及びFU競合と共に、メモリに関するデータハザードなど各種ハザードが並列的に解決されて、スケジューリングが行なわれるので、命令

の実行クロックサイクルの短縮化が計られ高速化が可能となる。

【0084】また本発明によれば、スケジューリング手段内に、命令の実行順序を逆にするることによる競合の有無を判別し、競合が有る場合にスケジューリングを遅らせる逆順序競合判別論理回路と、前記逆順序により競合が有るという情報に基づき該当命令を実行した結果情報のバイパスをキャンセルするバイパスキャンセル論理回路とを設けるようにしたので、誤った演算結果情報を得ることなく、Out-of-orderで複数命令を同時発行する装置を実現することができる。

【0085】また本発明によれば、前記パイプライン内の命令の流れを乱す各種ハザードを解決し、命令スケジューリングの対象となる命令間の実行されるべき実行クロック時間の差に着目したスケジューリング・パターン方式により作成した命令発行アルゴリズムを論理圧縮してPLAに実装することによって、Out-of-orderで複数命令の同時発行を行うハードウェア機構を簡易化できる。また、命令実行時には命令間のレジスタ競合データと命令種別データとがPLAに入力されると、前記命令発行アルゴリズムによって各命令の発行の可又は不可情報が出力され、パイプラインのクロックサイクルが短縮化されるので、演算処理の高速化が実現されている。さらに、機能ユニットを増やすなどユニット構成が変更されても、PLA内のアルゴリズムをソフトウェア的に変更するだけで、他のハードウェア要素をいっさい変更しないでよいという利点がある。

【図面の簡単な説明】

【図1】図2の命令デコード及びスケジューリング機構についての詳細な構成図である。

【図2】本発明に係るスーパースカラ・アーキテクチャを示すブロック図である。

【図3】本発明とDSのIDステージにおける処理方式の相違を説明する図である。

【図4】本発明に係るバイパス回路によるRAWハザード回避方法の説明図である。

【図5】本発明に係るWAWハザード回避方法の説明図である。

【図6】本発明に係るWARハザード現象の説明図である。

【図7】本発明に係る分岐命令を飛び越えてOut-of-orderが実行される場合の説明図である。

【図8】本発明に係るWARハザード回避方法及びその効果の説明図である。

【図9】タイプAのユニット構成の場合のPLA上の命令発行論理回路図である。

【図10】タイプBのユニット構成の場合のPLA上の命令発行論理回路図である。

【図11】タイプHのユニット構成の場合のPLA上の命令発行論理回路図である。

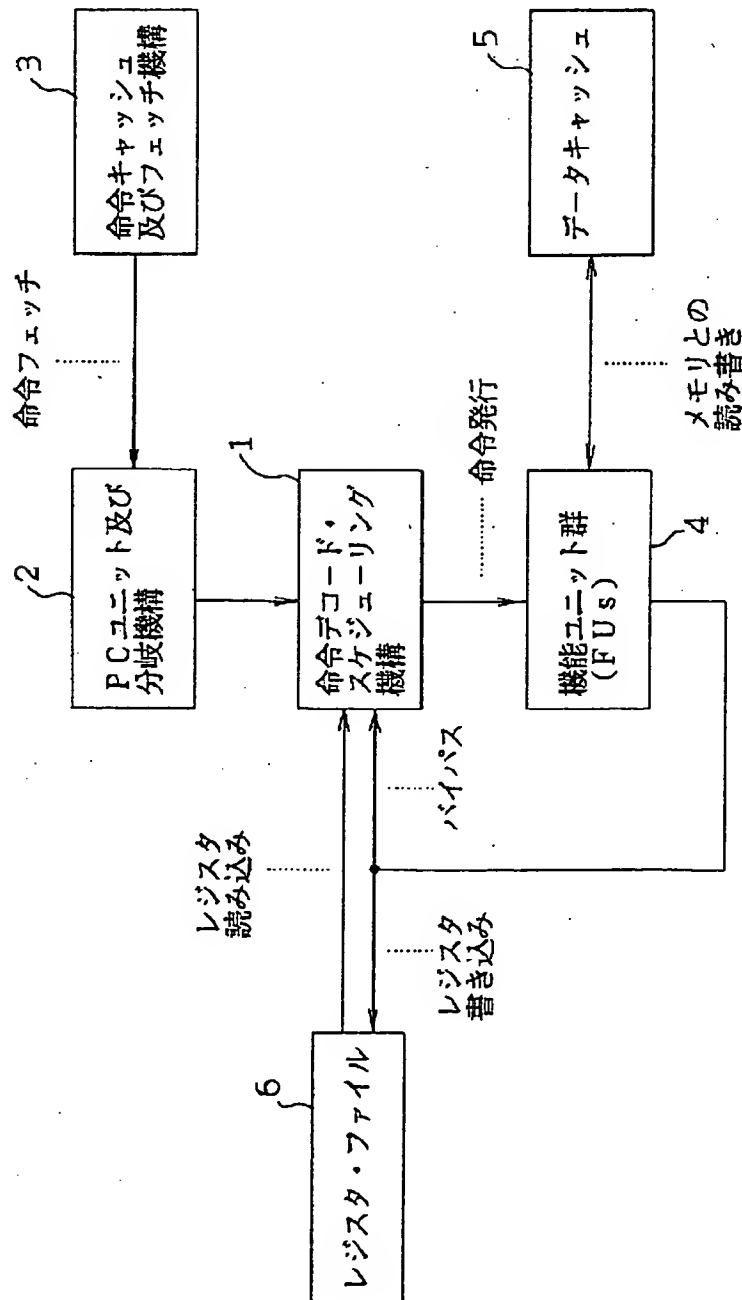
【図12】従来のRISCアーキテクチャのパイプライン構造を示す図である。

【符号の説明】

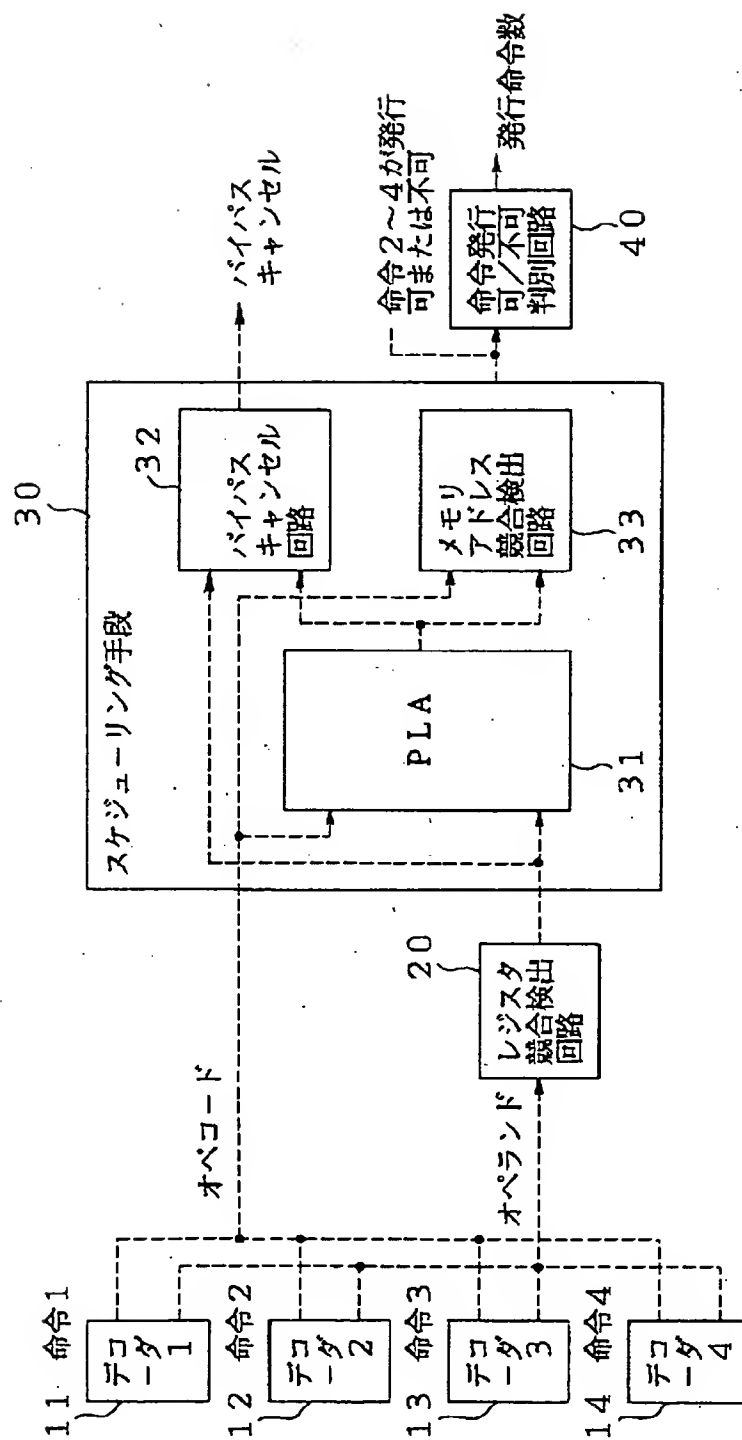
- 1 命令デコード・スケジューリング機構
- 2 PCユニット及び分岐機構
- 3 命令キャッシュ及びフェッチ機構
- 4 機能ユニット群 (FUs)
- 5 データキャッシュ

- 6 レジスタ・ファイル
- 11-4 デコーダ1〜デコーダ4
- 20 レジスタ競合検出回路
- 30 スケジューリング手段
- 31 PLA
- 32 バイパスキャンセル回路
- 33 メモリアドレス競合検出回路
- 40 命令発行可／不可判別回路

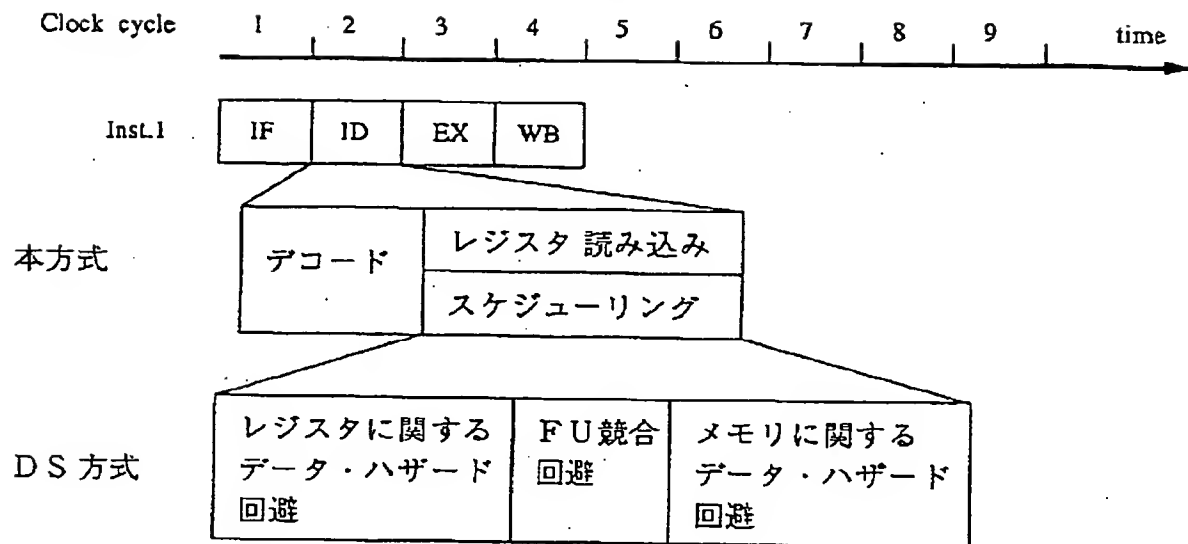
【図2】



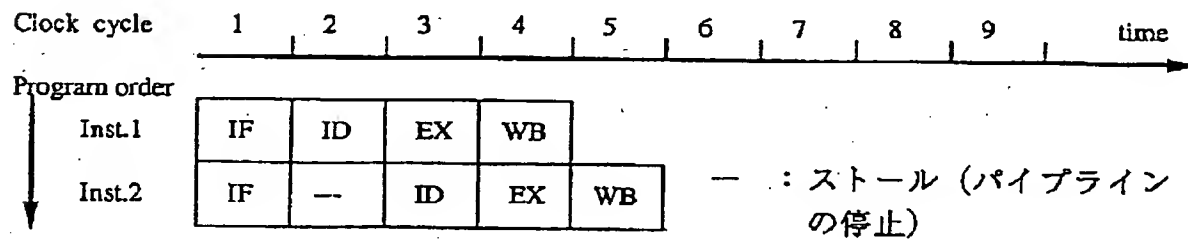




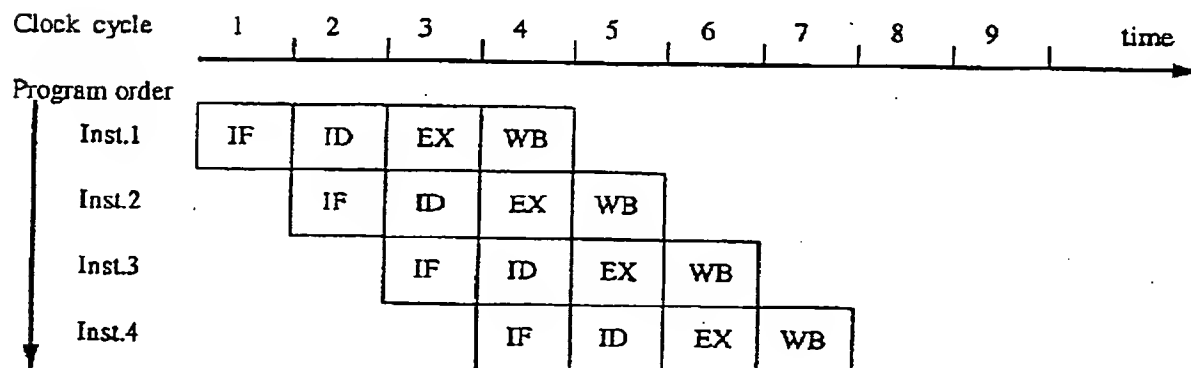
【図 3】



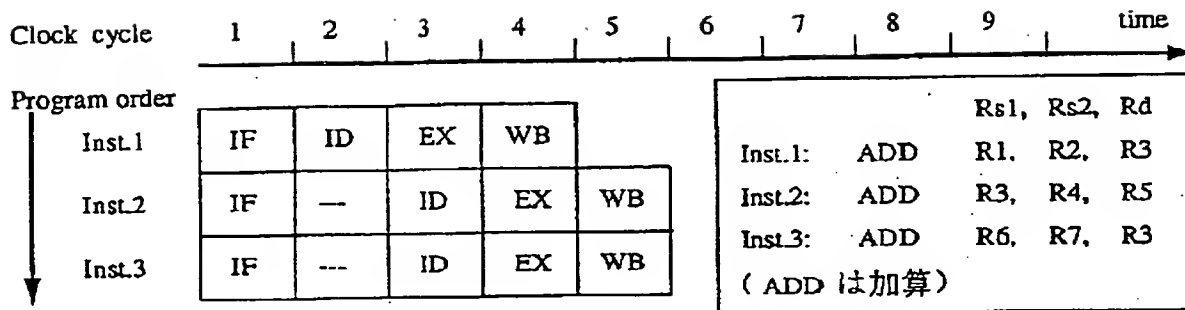
【図 4】



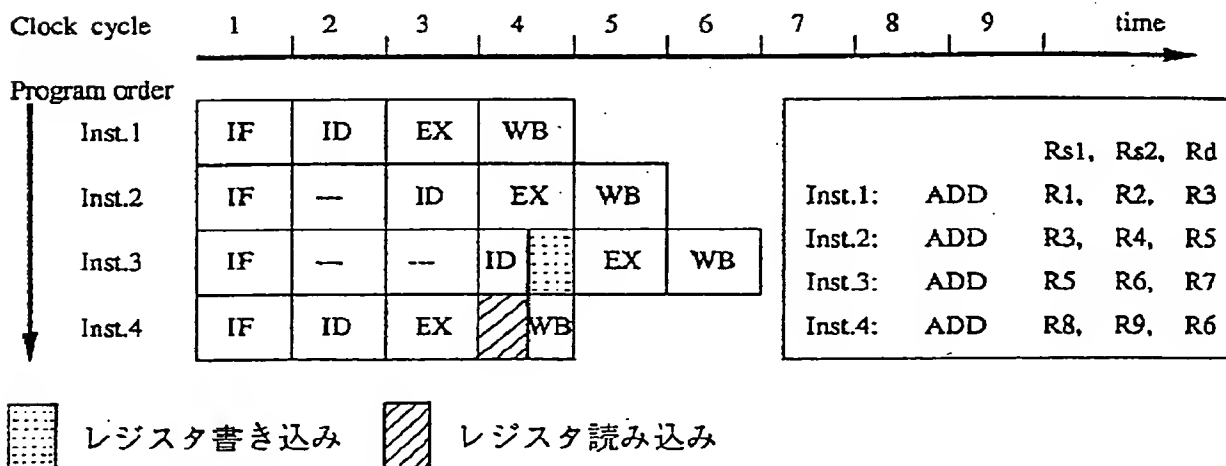
【図 12】



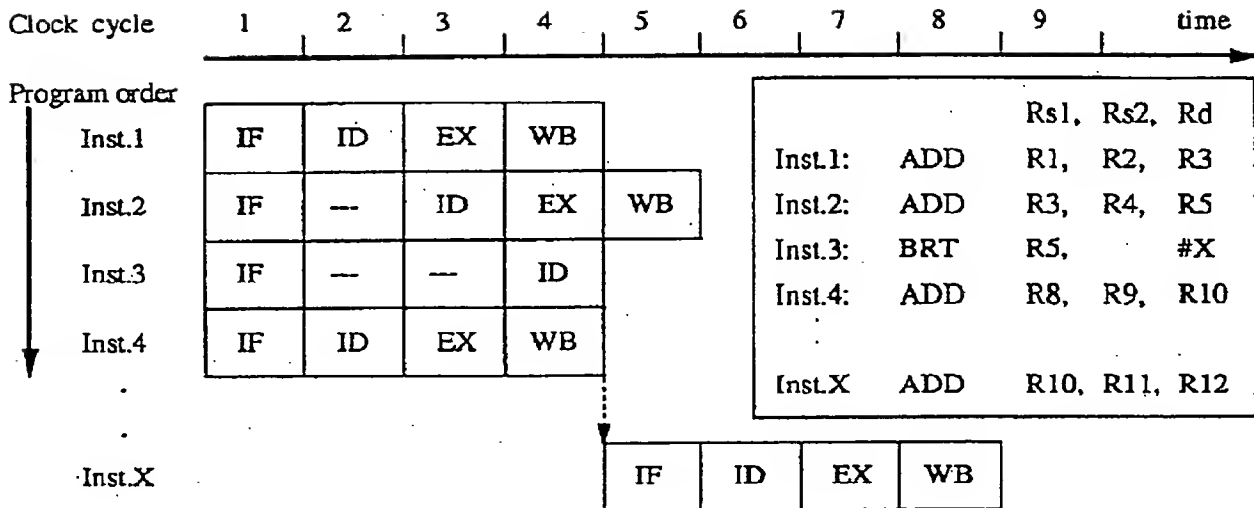
【図 5】



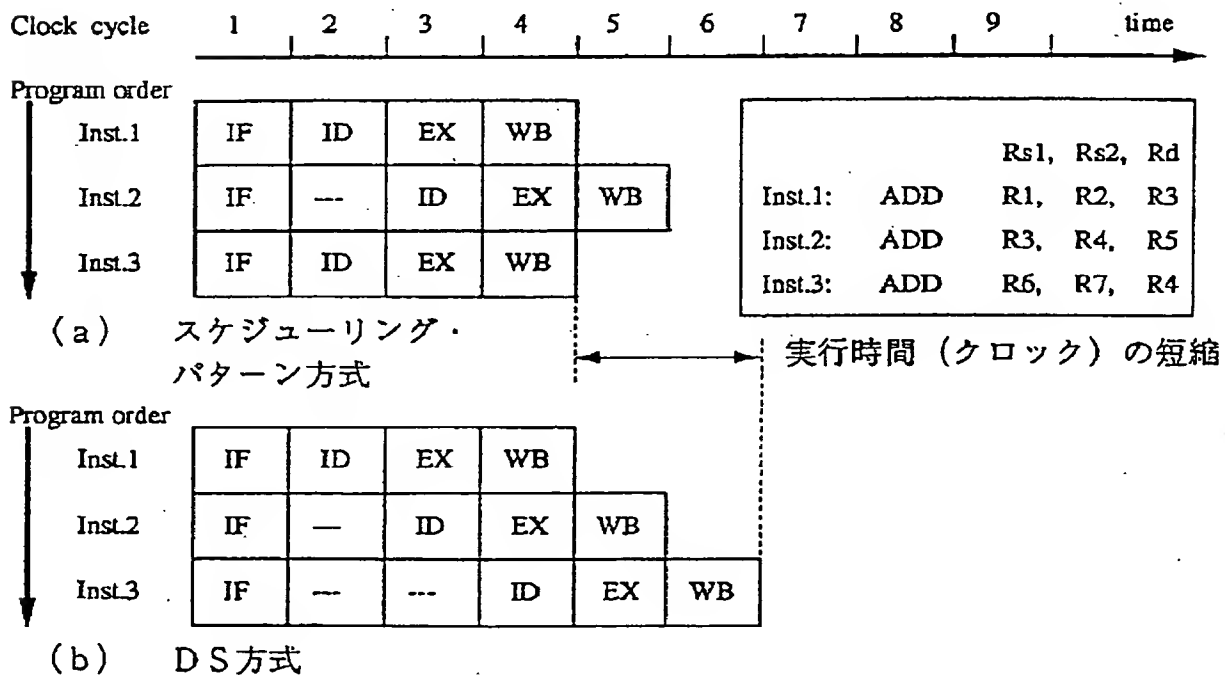
【図 6】



【図7】

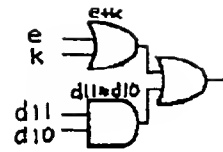


【図8】

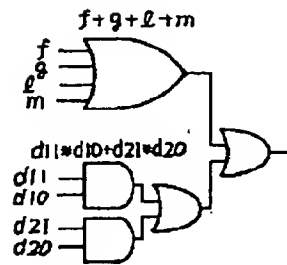


【図 9】

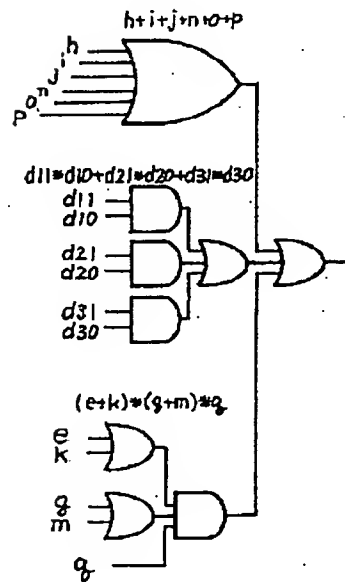
(a) 命令 2



(b) 命令 3

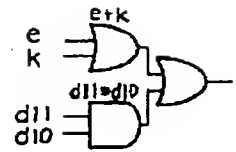


(c) 命令 4

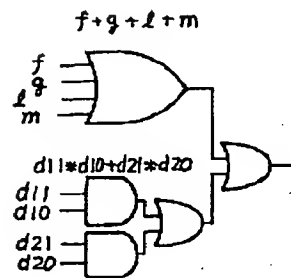


【図 10】

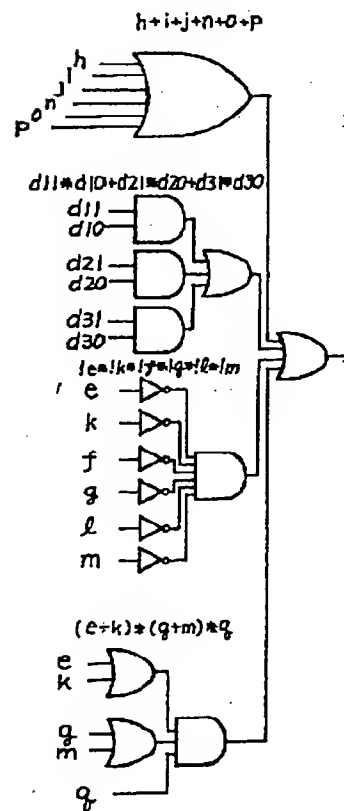
(a) 命令 2



(b) 命令 3

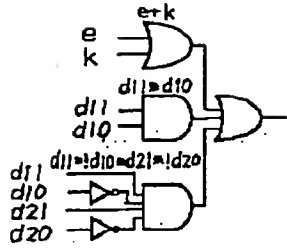


(c) 命令 4

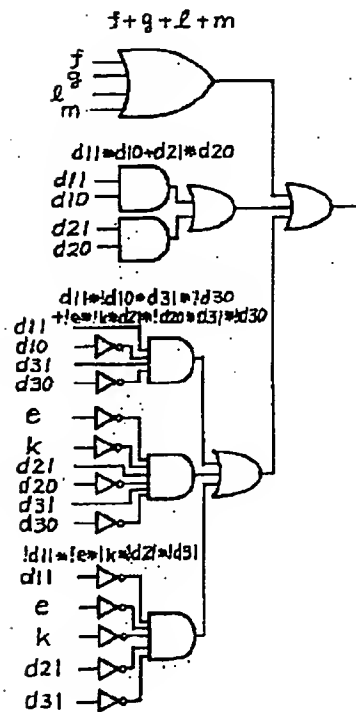


【図 11】

(a) 命令 2



(b) 命令 3



(c) 命令 4

